GEANT JRA1-T2-D7.3 Overview of SDN Pilots Description and Findings: Part B

Pavlidis, Adam; Mendiola, Alaitz; Kurbatov, Aleksandr; Papazois, Andreas ; Juszczyk, Artur ; Fernandez, Carolina; Conejo, David Garcia ; Taquet, Eduardo Juan Jacob; Roberts, Guy; Astorga, Jasone

Total number of authors:
24

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

25-06-18

Deliverable D7.3 Overview of SDN Pilots Description and Findings: Part B

Deliverable D7.3

Contractual Date:	31-12-2017
Actual Date:	25-06-2018
Grant Agreement No.:	731122
Work Package/Activity:	7/JRA1
Task Item:	Task 2
Nature of Deliverable:	R (Report),
Dissemination Level:	PU (Public)
Lead Partner:	GRNET
Document ID:	GN4-2-17-5EA76
Authors:	Adam Pavlidis (GRNET), Alaitz Mendiola (EHU), Aleksandr Kurbatov (GÉANT), Andreas Papazois (GRNET), Artur Juszczyk (PSNC), Carolina Fernandez (i2CAT), David Garcia Conejo (EHU), Eduardo Juan Jacob Taquet (EHU), Guy Roberts (GÉANT), Jasone Astorga (EHU), Jordi Ortiz (UMU), José Soler (DTU), Jovana Vuleta (AMRES), Kostas Stamos (GRNET), Lukasz Makowski (PSNC), Matteo Gerola (CREATE-NET), Pankaj Sharma (RENATER), Paolo Lungaroni (CNIT), Pier Luigi Ventre (CNIT), Sebastiano Buscaglione (GÉANT), Sharique Javaid (LRZ), Stefano Salsano (CNIT), Victor Olifer (JANET), Afrodite Sevasti (GRNET)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

Abstract

This document reports on SDN technology pilots utilising an infrastructure deployed across the GÉANT backbone in parallel to the infrastructure carrying production traffic. Technology pilots aim to verify the functionality and stability of novel, integrated software and hardware modules in a holistic way in an out-of-the-lab environment, while at the same time assessing the operational readiness of the SDN solutions. This deliverable updates deliverable *D7.1 Overview of SDN Pilots Description and Findings Part A*, and should be read as a standalone overview of pilot findings for all SDN use cases.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Task 2 of the Joint Research Activity 1 (JRA1) of the GN4-2 project has adopted the Software-Defined Networking (SDN) Use Cases prototyped during the previous GN4-1 [GN4-1 POC] project and transitions them to pilot status. The timeframe of this transformation lies within GN4-2.

This document reports on SDN technology intermediate pilot phases (Phase B) carried out by the JRA2 team, utilising an infrastructure deployed across the production GÉANT backbone, parallel to the infrastructure carrying production traffic. Technology pilots aim to verify the functionality and stability of novel, integrated software and hardware modules in a holistic way, in an out-of-the-lab environment, while at the same time assessing the operational readiness of the SDN solutions.

Following the work reported in deliverable *D7.1 Overview of SDN Pilots Description and Findings Part A*, [D7.1], this deliverable reports on the extended (in terms of feature coverage and scope) pilots of the developed SDN solutions (listed below), as well as pre-production activities, focusing on user-perceived functionality, usability and maturity of the solutions. Such activities focus on eliminating issues and adding functionality following the updates of the utilised software and vendor hardware. Pilot results are constantly updated, based on rapid codebase/hardware feature evolution.

This deliverable updates *D7.1 Overview of SDN Pilots Description and Findings Part A*, and is meant to be read as a standalone overview of pilot findings for all SDN use cases.

The SDN use cases include:

- **SDX L2**: This use case provides software-controlled configuration of point-to-point Layer 2 services between access (Ethernet) interfaces of interconnected parties, by introducing SDN capabilities to GÉANT Open [GOPEN] for R&E parties or equivalent NREN facilities, thus removing the need for the operators to manually configure the fabric (i.e. operators do not have to configure directly on the switching equipment). Exchange points can thus become programmatically configurable by the interconnected entities, under certain isolation principles.
- **SDX L3 / SDN-IP**: This use case deals with Layer 3 capabilities within a software-defined exchange (SDX) context, where an SDN controller acts as a route server on behalf of the member autonomous systems (ASes) at the exchange point. It also deals with the SDN IP application, allowing IP transport over an OpenFlow-capable network for connected external Border Gateway Protocol (BGP) routers and their ASes.
- **SDN-Circuit on demand**: The provisioning of circuit-on-demand services utilising SDN-based flow handling, combined with NSI protocol support for multi-domain discovery and signalling is the aim of this use case. It also adds advanced path computation, enhanced resiliency and agile flow-handling capabilities to existing services of GÉANT/NRENs [NSI].

- **Transport SDN**: The aim is to achieve software-controlled capacity allocation at the optical layer of GÉANT (based on the Infinera platform). This can fit the needs of elephant flows and route bypass as/when needed to selectively reduce costs of routing traffic.

At the time of writing, only the SDX L2 and SDN-Circuit on demand use cases are being further developed as pilots. The initial transport SDN testbed was not chosen to be evaluated as a pilot due to the cost of the optical platform on which it was developed and the expected transition from it. As explained in Section 5, SDX L3/SDN-IP was also not chosen to transition to pilot due to software scalability concerns from the initial testbed. The SDN-based circuit on demand was evaluated as a pilot in GÉANT, detailed in Section 4, and also in a Janet testbed consisting of two domains, detailed in Appendix A, and in the context of an EHU experimental facility leveraging Network Function Virtualization and SDN, detailed in Appendix B.

The SDN controller ONOS has served as a basis for the implementation of the use cases [ONOS]. Its modular architecture allows for the pluggable extension with applications that enable the ONOS core to communicate with extended functionality at both the northbound (applications) and southbound (drivers) layer. For this purpose, the JRA1 Task 2 team continued its close cooperation with the ONOS community and the Open Networking Lab (now merged in the Open Networking Foundation [ONF]).

During Phase A, the use cases were prototyped in a lab environment, using a variety of OpenFlow-enabled equipment, including Corsa 6400, Pica8 and Dell switches. Previous generation Corsa switches used at the start were gradually replaced with a newer Corsa DP2000. Their SDN data plane supports switch virtualisation and is the platform deployed for the SDN technology pilot environment.

The Corsa DP2000 platform allows for full virtualisation of the hardware, where all the hardware resources of the switch can be exposed as independent virtual SDN switches or routers. Such an approach enables division into Virtual Forwarding Contexts (VFCs) that appear as independent switches to the OpenFlow controller with variable pipeline characteristics. Thus, advanced OpenFlow characteristics are maintained in this version of the hardware, with different multi-table pipelines now made available on a VFC basis rather than a switch basis. For the proper operation of the use-case applications, the appropriate ONOS driver had to be used for each VFC, in order to manage flows on a per-VFC basis, as best practice for multi-table pipeline switches supported by OpenFlow 1.3 denotes. The Corsa platform supports several VFC types, including the "openflow" VFC type, which presents a single table and therefore does not require a specialised ONOS driver.

In close collaboration with SA1, the SDX L2 and SDN-based, circuit-on-demand pilots have been further elaborated during Phase B, and the outcomes are presented in detail here.

2 Pilot Assumptions

An important, overarching objective of the pilots is to assess the resiliency of the control plane. Because SDN brings critical functionality out of the network equipment and into software components, the SDN controller needs to be robust enough in response to various types of network, equipment and software failures. ONOS promises to deliver pioneering clustering functionality, which the pilots are designed to verify. Another critical and relevant aspect is the assessment of monitoring operations and capabilities that need to be provided in the SDN use-cases, at least on par with the current operational practices.

2.1 Infrastructure

SDN technology pilots have been implemented on an overlay network that GÉANT Operations has deployed using Corsa DP2100 switches. This network is shared with JRA2 and the GÉANT Testbeds Service (GTS), which takes advantage of the virtual switch capability and has access to the administrative underlay of the switches. JRA1 access is limited to the overlay (VFC – virtual instances), which are provisioned manually.

The SDN pilot infrastructure, based on Corsa white box switches, is interconnected by 10Gbps lambdas and is deployed in London, Paris, Amsterdam, Prague and Milan. Each location includes one Corsa DP2100 box, which is shared with the GTS infrastructure. Figure 2.1 below shows the underlying physical infrastructure of the GÉANT network that has been used for the deployment of SDN pilot slices.

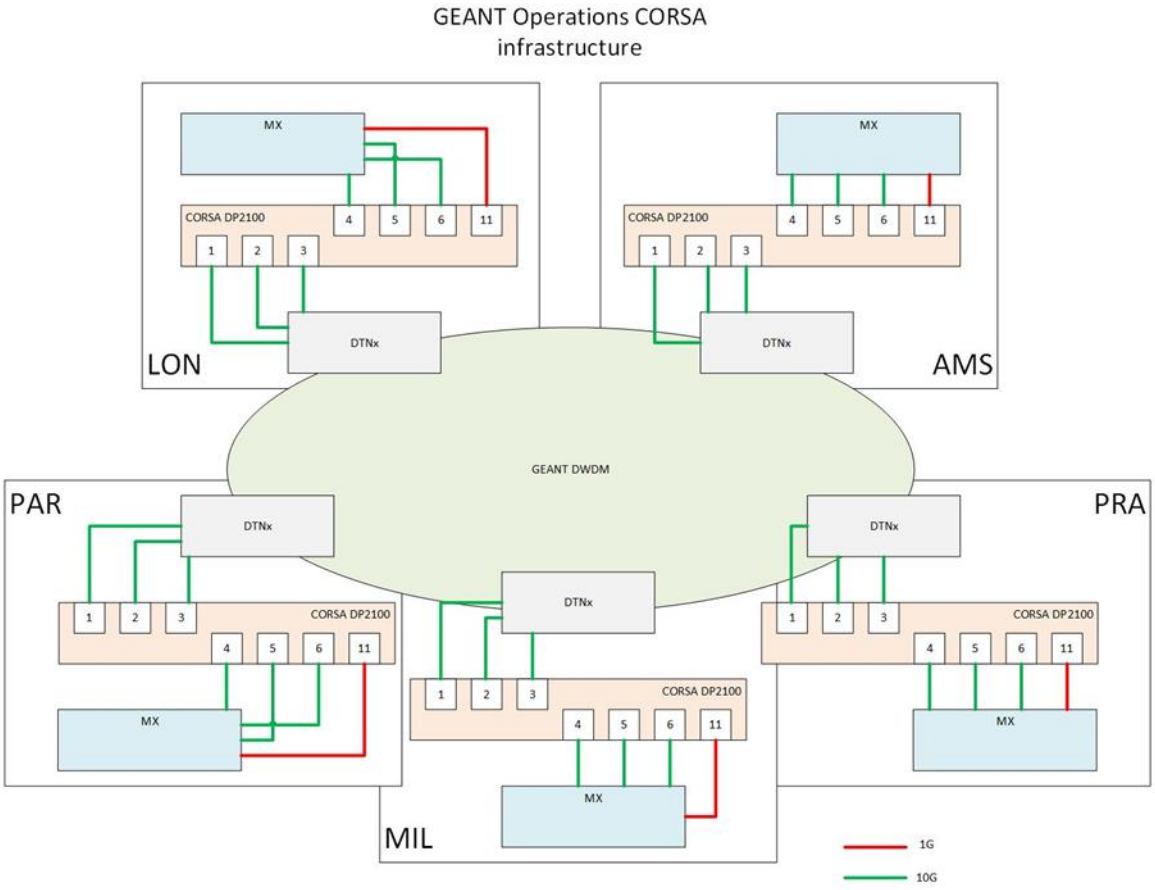

Figure 2.1: Physical infrastructure

Virtual Forwarding Contexts (VFCs) on Corsa boxes are OpenFlow enabled, and a dedicated L3VPN is used for connectivity with the controller. Users gain access to the data plane of the SDN pilot infrastructure via their normal interconnect at their local GÉANT MX router.

2.2 Stakeholders

Pilot outcomes are important for GÉANT and partner NRENs in GN4-2. During the first 12 months of the project, consultation with participating members from NRENs took place in order to identify the overall characteristics that are considered important by NREN operators in this context. Detailed feedback from this process is provided in the following sections.

3 SDX L2 Pilot

3.1 Introduction - the SDX L2 Concept

GÉANT provides GÉANT Open [GOPEN], a connectivity service for NRENs to connect with external (non-GÉANT) networks through Open eXchange Points (OXPs). Inside an OXP, the users (NRENs or other non-NREN external participant networks) request the establishment of Layer 2 circuits between end-points, which are manually provisioned through VLAN tunnels, according to the relevant GÉANT Open Exchange Policy rules. Several NRENs and regional networks currently operate similar services.

In order to set-up a GÉANT Open service between two access points (ports or VLANs) inside an OXP, the user has to contact the operators to manually configure the connection. These operations (creation of virtual interfaces, VLAN ID selection on both endpoints, VLAN ID rewriting) are error-prone, and require coordination between the interested parties. Any arising issue also requires further manual intervention from the operators. A typical target for the provisioning time of these services is five days. In addition, as some of those exchanges are in the process of being interconnected, the waiting time to establish a service between two end-points increases to at least about ten days. In general, this results in a lengthy, cumbersome and manual process for setting up new services.

This process could be streamlined by the use of the SDX-L2 service, the SDN-isation of the GÉANT Open service. Using SDX-L2, the provisioning process is reduced from days to minutes. Moreover, by leveraging features provided by SDN controllers, most failure cases are automatically resolved without any manual intervention: a failure of a controller within a cluster is solved using redundancy of SDN controller instances, and a data-plane failure is solved with automatic re-computation of data plane paths around the faulty network element, pending capacity and adequate protection foreseen by the service.

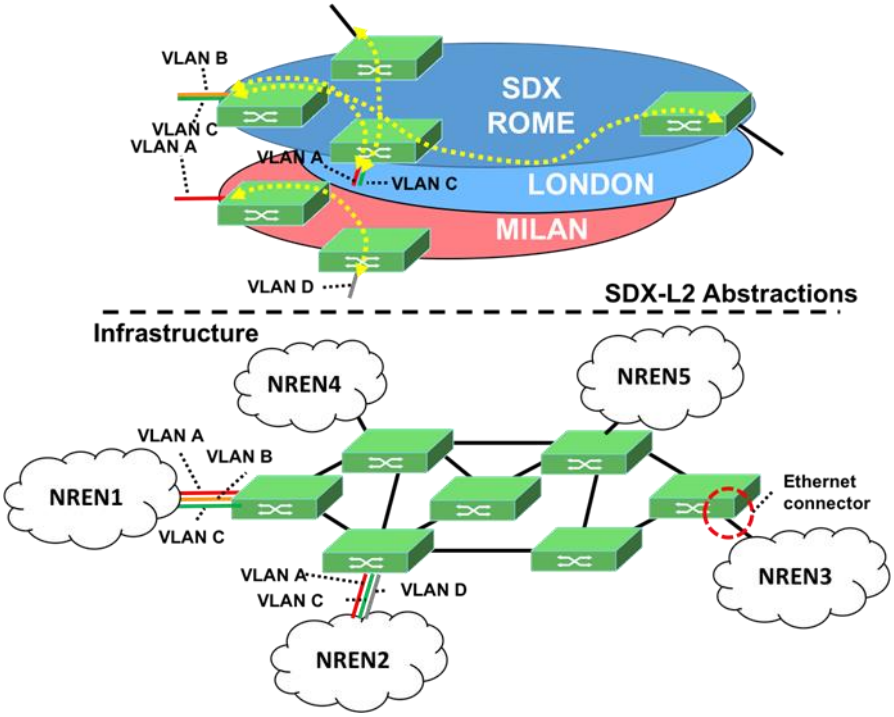

Figure 3.1: SDX-L2 application and its abstractions

SDX-L2 provides users with powerful abstractions, as shown in Figure 3.1. A virtual SDX (e.g. Rome, London or Milan) contains a number of edge ports or VLANs (untagged or tagged ports from physical devices), which are internally modelled as edge connectors. Network operators can establish virtual circuits between these ports, as required, to serve each individual customer. Alternatively, customers that operate only on the edge of the SDN network are controlled by SDX-L2. This procedure eases service management and provisioning by reducing provisioning times, providing easier abstractions and operations to network operators, enforcing isolation and avoiding several types of conflicts during service creation.

3.2 Description

Following Phase A of the SDX L2 pilot, GÉANT Operations has further assessed the technology pilot on the GÉANT operational network based on pilot findings addressed by the JRA1 Task 2 team, as well as additional features. For this phase, ONOS master release (1.12 at the time of testing) has been used, as it contains a thin driver developed by the JRA1 Task 2 team and was required to mitigate a 'clear actions' instruction originally sent by ONOS to support the Corsa DP2100 devices (as detailed in 3.7.1). Regarding Corsa software, Corsa DP2100 - 2.1.5 build 8 has been used.

The features of the technology pilot and their limitations during Phase B are clearly described in the following sections. The detailed set-up of the pilot infrastructure and supported features, as well as the SDX-L2 application architecture can be consulted in Sections 3.5 and 3.6.

3.3 Functional/Operational Overview

The functional/operational features of the pilot include both the existing capabilities of the operational exchange points and the new functionality introduced by the proposed solutions.

Below is a summary of basic capabilities:

- Definition of L2 connections between two edge ports or VLANs.
- Definition of multipoint L2 connections between edge ports or VLANs.
- VLAN and Stacked-VLANs (802.1ad) encapsulation for the circuits.
- [*] MPLS encapsulation for the L2 connections.
- IPv6 support.
- Control plane resiliency.
- Control plane failure recovery.
- Network status after control plane failure.
- Traffic re-routing after data plane failure.
- Allow collecting and storing statistics related to traffic and errors and polling by using SNMP from other systems.

- Allow sending SNMP traps to central monitoring systems, advising on the status of interfaces and L2 connections across virtual switches in a single domain. Ideally, this should also provide the status of the overall service across multiple domains.
- Provide logging facilities, with the ability to distinguish system-related and network-related events. Logging facilities should be easily accessible for support teams, also logging on to separate remote servers should be possible.
- Detailed information about services configured across multiple domains should be accessible (i.e. connection points, capacity, VLANs, start/end times) in a standard format, such as REST, to allow external recording or further processing. A specific use case is a Configuration Management Database (CMDB) and incident reporting.
- The controller should allow for easy access to information about interfaces and services. This should be presented in a format that can be read by NOC staff. It should not take a number of commands to obtain traffic, errors, interface state, speed, configured VLANs and interface descriptions information for one or multiple interfaces on the virtual or physical switches. This information could be collated and presented using troubleshooting facilities via the GUI if not available in this format through the command line interface (CLI).
- Allow diagnosing root cause, or as a minimum, locate the sub-system(s) where the fault originates: controller, orchestrator, switch.

  [*] ONOS controller and SDX-L2 support this, however, at the time of writing, Corsa devices did not.

Further desired functionalities include:

- Isolation of virtual switches and related controllers. Issues on a virtual switch and its controller should not have any adverse impact on any other virtual switches, their controllers or the physical box itself.
- Allow part of a virtual switch to control a specific range of VLANs on a port to allow a single physical connection to be part of several virtual switches.
- Allow rate limiting VLANs on interfaces for defined bandwidth services.
- Add and remove ports and/or VLAN ranges from a virtual switch without impact on other ports, services or VLANs on the same switch.
- Provide an API for integration of the domain controller or multi-domain controller with other systems, as well as support for NSIv2.
- Allow, at the controller side, for the increase of the number of switches under a controller without fully re-configuring the whole domain and associated services. A serviceable interruption during the expansion of the fabric is acceptable.
- Allow for a hierarchy of users and admin accounts, including fine-grained authorisation for access to particular ports or VLANs per user or user group, as well as a clear definition of rules for resource management by multiple owners.
- Support of LAG with link aggregation control protocol (LACP). Functionality of the LAG interfaces must be identical to the physical interfaces in terms of configuration options and the report options (show commands, statistics, etc.)

The latter group of requirements extends beyond the core SDX L2 concept, and is subject to an integrated SDN-ised L2 service offering.

3.4 Pilot Overview

In the context of JRA1, two ONOS applications have been selected as suitable to fill most of the described requirements, SDX-L2 [SDX-L2] and VPLS [VPLS]. SDX-L2 was developed within JRA1 with the aim of offering an implementation of the SDX L2 use case concept and offered to the ONOS community, whereas VPLS has been developed by ONOS and JRA1, who are contributing to its further development and testing. The main difference between SDX-L2 and VPLS is that VPLS offers the capability for creation of multi-point connections, whereas SDX-L2 is limited to point-to-point connections only.

During Phase A testing, the available release of the Corsa firmware used in the pilot environment, did not include some features, such as multi-point connections via the VPLS application. Therefore, SDX-L2 was the first application to be deployed in the technology pilot over the GÉANT network.

The following list contains the functional requirements that have been included in the technology pilot, including Phase B :

- GUI-based management, featuring SDX-L2 features and highlighting VCs on topology.
- Some monitoring systems and operational tools to quickly assess the status of the cluster and to backup logs.
- IPv6 support.
- Definition of L2 connections between two edge ports or VLANs.
- [*] VLAN and Stacked-VLANs (802.1ad) encapsulation for the circuits.
- Control plane resiliency.
- Control plane failure recovery.
- Network status after control plane failure.
- Traffic re-routing after data plane failure.
- The controller should provide an open API to access management and service information, such as traffic, errors, interface state, speed, configured VLANs and interface description information for one or multiple interfaces on the virtual or physical switches. It is expected that the controller will eventually be complemented by an application tailored for NOC use.
- Allow diagnosing root cause, or as a minimum, locate the sub-system(s) where the fault originates – Controller, orchestrator, switch.

[*] Only VLAN circuits were tested, Stacked-VLANs are currently not implemented in the ONOS Intent framework.

Details about the pilot functionality are reported in the following sections, along with a short summary of functionality testing results. During the testing of the SDX-L2 application, the GÉANT operations team encountered a number of limitations. The most serious ones relate to the operational perspective:

- The availability of remote monitoring of the cluster is limited (e.g., not available via SNMP or not very granular) [ONOS-CLUSTER].
- Sequential VLAN tag selection on the trunk is not always ensured. This can happen under network errors (like in link flapping), where the topology will continuously vary. The controller attempts to

internally regenerate the intents in case of failure. This may sometimes trigger the usage of non-sequential VLAN tags; as the previously assigned VLANs may not be available as the ONOS Resource manager is not able to release the previous allocation upon failure.

- In some circumstances, it is possible that after recreating some intents through the VCs, these (and the corresponding flows) get stuck and can no longer be fully deleted.
- The internal resource tracking can fail, leading to the lack of discovery of some physical resources (devices and their ports), or missing registration of virtual resources (VLANs and bandwidth).

3.4.1 Limitations Imposed by the Pilot

Some technical limitations exist and are imposed by the nature of the configuration and setup of the pilot slice. Such limitations are mainly affecting hosts-to-VFC connectivity as hosts are deployed on the GÉANT IT infrastructure.

The main limitations of the slice are:

- Due to the VM infrastructure used to instantiate the end hosts, they cannot use VLANs when connecting to Corsa VFCs.
- Due to the connections to the VFCs, hosts can send less than 100Mb to the VFC.
- Only 50% of the table space can be assigned per pipeline type to VFCs to be used for testing, due to simultaneous usage of the VFC resources in the GÉANT network.
- Links between the Corsa DP2100 in different sites need to support traffic for multiple testbeds and therefore cannot have the capacity fully dedicated to a specific slice. This means that a limit of 1G is to be considered for all links to VFCs.
- Only part of the pilot is configured to transmit across edge hosts.

The limitations are imposed by the nature of the IT infrastructure where the VMs are deployed, as such infrastructure is built for supporting web services available through the global R&E internet and not a virtualised slice for testing. As a workaround, MX routers' sub-interfaces can be used to test using tagged traffic, however, the number of VLANs that can be configured on MXs for simultaneous use are limited to 20. In addition, only simple, non-volumetric tests can be run between MXs sub-interfaces. Also, not all intermediate routers and end hosts may be fully configured.

The limitation on VFCs and Corsa DP2100 connectivity is due to the shared pilot facilities usage, i.e. the fact that the same physical switch has to support multiple use cases and GTS needs, therefore, the switch and its links cannot be fully dedicated to testing for a single use case.

3.4.2 Features and Functionalities Not Available

Some features were still not available at the time of Phase B pilot testing.

The multipoint connections were not included in SDX-L2. Also, the MPLS encapsulation on trunks is supported by ONOS and SDX-L2, but there are not any known future plans from Corsa to add support for MPLS labels.

3.5 Detailed Infrastructure Setup

A network slice has been provisioned in the production GÉANT network to provide an operational environment for the SDX-L2 Pilot. Figure 3.2 illustrates the environment.

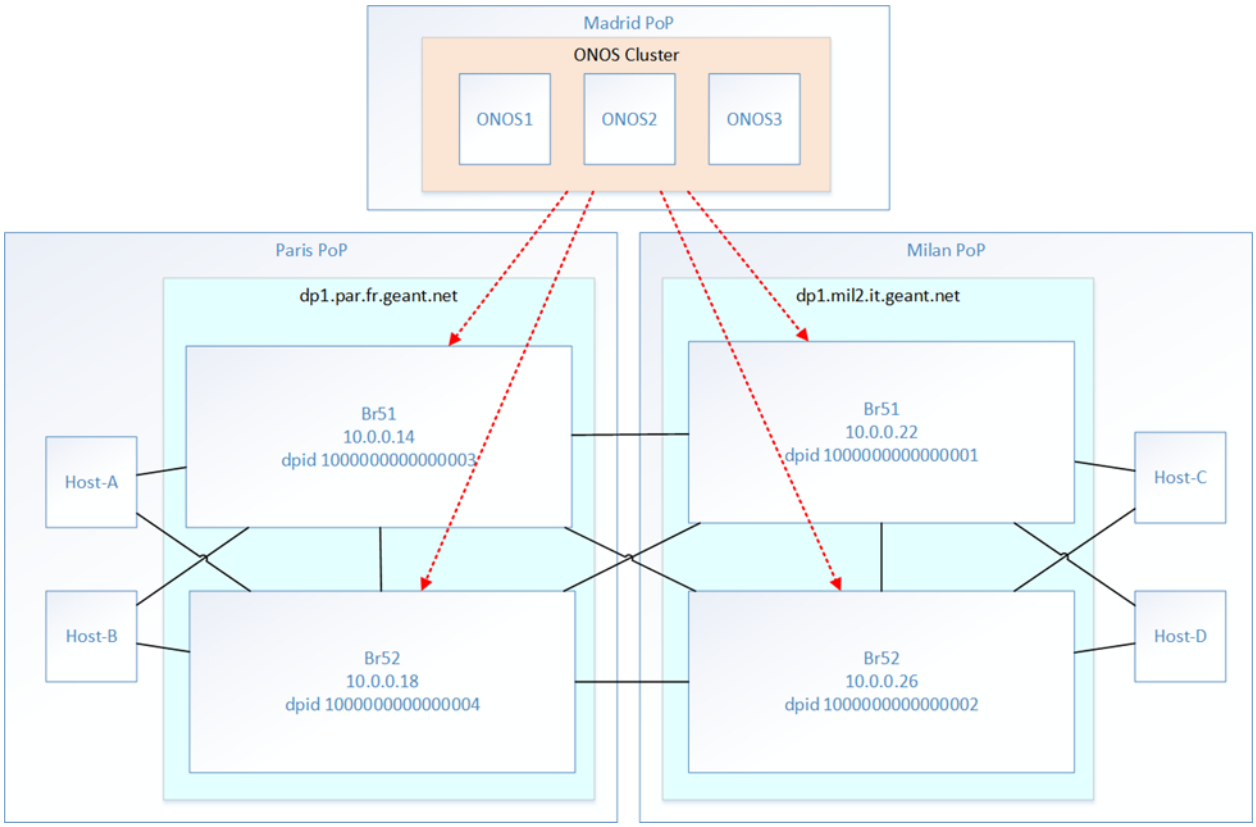

Figure 3.2: SDX-L2 pilot

A set of four VFCs have been configured on two Corsa DP2100, deployed in the GÉANT PoPs of Milan and Paris. VFCs are interconnected using a mix of Infinera and Juniper MXs-based point-to-point links.

Seven VMs have also been provisioned: Three to function as a set of ONOS instances in a cluster (ONOS1, ONOS2, ONOS3) and four to be used as end hosts (HOST-A, HOST-B, HOST-C, HOST-D). Hosts are deployed in the GÉANT IT VM infrastructure local to the Corsa switch, where hosts are connected.

Connectivity between the IT VM infrastructure and Corsa DP2100 is provisioned across local LAN switches and MX routers. An L3VPN has also been configured in the GÉANT IP/MPLS network to provide connectivity between the ONOS cluster and VFCs while maintaining isolation from the rest of GÉANT IP network.

3.6 Detailed Software Architecture

SDX-L2 provides operators with high-level APIs:

- Operators see the abstraction of managing virtual SDXs.
- An SDX contains edge connectors; either physical or virtual. These can be tagged (VLAN-tagged interfaces) or not (device edge-ports).
- Connectors can be interconnected through virtual circuits (VCs). The VCs can be encapsulated (VLAN, MPLS) or not.

Different sub-components are in use:

- SDXL2Service: allows creating the SDX, which can later on contain VCs.
- SDXL2VCService: allows creating the VC, coupling two different connectors.
- SDXL2MonitoringService: performs a background check on failed intent installations and on failure regarding edge-ports.

The rest of subcomponents provide access to the operator (through CLI and GUI), register information, or interact with the core to manage network resources (IntentService, PacketService, EdgePortService).

ARP and NDP packets are not forwarded in the data-plane, but instead are relayed through the control plane by using the SDXL2ARPNDPHandler sub-component.

The current architecture of SDX-L2 is depicted in **Error! Reference source not found.**.

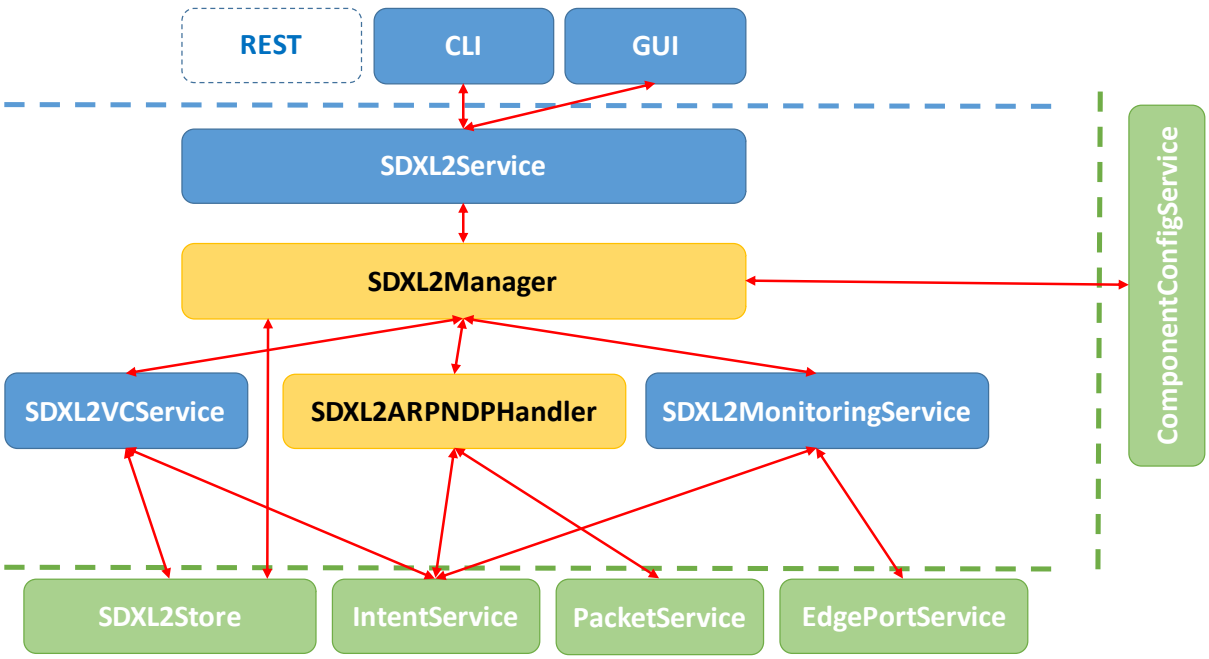

Figure 3.3: SDX-L2 architectural components

Initially, the architecture allowed the creation of the following types of virtual circuits:

- MAC-based: Point-to-point communication would be achieved by registering the MAC addresses of the endpoints.
- VLAN-encapsulated: a VLAN tag is defined per connector. If different VLANs are used per connector, VLAN translation will be performed.
- MPLS-encapsulated: An MPLS label is defined per connector.

Results obtained from SDX L2 pilot Phase B are detailed below.

3.7 SDX L2 Pilot - Detailed Results

Detailed results obtained from SDX L2 pilot Phase B are detailed in the SDX-L2 pilot results appendix located on the GÉANT project Wiki. Table 3.1 below summarises the result per feature tested in Phase B, and the expected reason for the failing conditions, where applicable.

Functionality group	ID	Requirement	Result	Remarks
SDX-L2: definition of SDX data	1	Creation of an SDX instance	Success	
	2	Deletion of an SDX instance	Success	
	3	Addition of connection points	Success	
SDX-L2: creation of circuits	4	Creation of a circuit – no access VLAN, no encap, no transit	Success	
	6	Creation of a circuit – access VLAN hostA to no VLAN hostC, no encap, no transit	Success	
	7	Creation of a circuit – access VLAN both end, NONE encap, no transit	Success	
	9	Creation of a circuit – diverse access VLANs, no encap, no transit	Success	
	11	Creation of a circuit – diverse access VLAN, NONE encap, transit	Success	
SDX-L2: creation of circuits with encapsulation	5	Creation of a circuit – no access VLAN, VLAN encap, no transit	Partial	Expected: first available VLAN on trunk Obtained: not necessarily the first available VLAN on trunk
	8	Creation of a circuit – access VLAN, VLAN encap, no transit	Partial	Expected: preserve VLAN tag on egress Obtained: VLAN tag is kept on output Packets. No connectivity – intermediate devices need to be configured

Functionality group	ID	Requirement	Result	Remarks
	10	Creation of a circuit – diverse access VLANs, VLAN encap, no transit	Partial	See 8
	12	Creation of a circuit – access VLAN, VLAN encap, transit	Partial	See 8
	13	Creation of a circuit – diverse access VLANs, VLAN encap, transit	Partial	See 8
SDX-L2: removal of data	14	Deletion of circuit	Success	
	15	Removal of SDX and all circuit are removed	Success	
	16	Removal of host and all associated circuit are removed	Success	
	17	Clean state	Success	
Controller & cluster behaviour	18	ONOS host reboot	Partial	Circuit is still working but there were issues getting ONOS automatically back to state before reboot
	19	Check ONOS cluster status	Success	Single command to obtain cluster status and access.
	20	VC removal from different ONOS VMs	Success	Expected: Remove VC from ONOS2, check the result in Corsa bridges Obtained: VC no longer in bridges
	21	Connect to ONOS application from ONOS2 and ONOS3 VMs	Success	ONOS is reachable with the same command from the different ONOS VMs

Table 3.1: SDX-L2 test results summary

Three main error conditions were observed:

1. The VLAN assigned to use encapsulation on a VC can be defined to be sequential, yet this is not ensured. Error conditions on the network may trigger recalculation of the intents, yet it can happen if the next available VLAN is not free. Moreover, in some cases, the release of the resources can fail, which contributes to unavailability of non-sequential VLAN IDs available.
2. The recreation of intents may reach an unstable status, where some get stuck and can no longer be fully deleted.
3. The internal resource tracking can fail, thus not registering physical (devices, ports) or virtual (VLANs, bandwidth) resources.

The possible steps to deal with such conditions are discussed below:

1. Configuring ONOS to provide the first available VLAN for VCs and ensure there are no continuous network errors (e.g. flapping links). ONOS Resource manager needs to be fixed when releasing resources.
2. No clear solution is available at the moment. Investigation should start with checking possible incompatibilities with the key name used for such intents.
3. Ensure that resources are registered during the setup of the cluster.

3.7.1 Blocker Conditions and Proposed Solutions

In addition to the results of the functional testing described above, there have also been some conditions on the ONOS applications and the network that have introduced issues in the technology pilot. Descriptions of the issues faced as well as the proposed solutions for Phase B of the pilot are detailed below.

3.7.1.1 *Issue with ONOS VPLS App*

During Phase A, VPLS translated multi-point connections into multiple output ports for a specific flow. However, this behaviour was not supported in the Corsa switches. In Phase B, Corsa released a new firmware version that supported multiple output actions in its future firmware releases. Consequently, this issue was solved and VPLS app was later successfully tested in the Cambridge lab verifying the issue rectification.

3.7.1.2 *Random failures*

During Phase B, it was observed that the controller was not able to converge to a stable topology. In fact, this resulted from continuous perceived errors in the underlying topology. Tests were carried out, both from GÉANT Operations engineers (the GÉANT team), to determine whether link failures were detected across devices, and from the SDX-L2 team (to determine whether this could be mitigated). To test link failure detection, the GÉANT team left scripts running that transmitted a number of packets, then analysed the ratio of successfully transmitted data. Since that approach found no errors, the SDX-L2 team attempted several tries to i) modify the link detection in the ONOS core and ignore some of them, given a certain minimum threshold, and ii) configure the controller to define values such as the maximum number of milliseconds between events (DefaultTopologyProvider's *maxIdleMs*) or the frequency of sending LLDP probes for port detection (LldpLinkProvider's *probeRate*). None of these efforts led to a final, 100% effective solution. The random failures completely disappeared after a couple of weeks, which did not give the testing team the chance to identify the exact cause.

3.8 Findings and Future Work

The main users of the technology pilot, the GÉANT team consider that the SDX-L2 application and the ONOS Intent platform are not ready for use in the production network. Most of the limitations reported during the evaluation of the pilot in Phase A (mentioned in Section 3.4) were addressed

during Phase B by the SDX-L2 team working in JRA1 T2. The limitations reported during the evaluation of the pilot in Phase B have been investigated and documented here for reference.

The SDX-L2 GUI has been available for the Phase B pilot. This version includes the L2 circuit GUI, which highlights the specific Virtual Circuits as an overlay in the topology depicted in the ONOS GUI. Missing features relate to better error handling, as well as improving the manageability of the creation of resources to make use of panels rather than tables.

In Phase A, JRA1 T2 team started the development of a new Corsa driver for ONOS in order to make Corsa switches compatible with last versions of ONOS (i.e., accounting for “deferred actions” which were not understood by the devices). In Phase B, Corsa provided an updated firmware that implemented support for the “clear deferred actions” instruction, multiple output instructions and OpenFlow “ALL” group. That meant that the VPLS app was made available for its pilot tests. Moreover, with the support of the ‘’clear deferred actions” instruction, custom drivers for ONOS are no longer needed.

The development work performed to update SDX-L2 during Phase B includes the improvement of the GUI to highlight the circuits in the topology and the improvement and refactoring of the core logic of SDX-L2 (such as a revisit to the ARP proxy, simplifying and hardening the managers or updating the generation of intents to match that of the latest ONOS).

The work developed during Phase B also includes:

- GÉANT OF conformance scripts to be run at each release of new Corsa firmware against the hardware switches.
- Automatic testing solutions based on Ansible scripts to assess proper behaviour of the SDX-L2 features.
- Automatic deployment solutions based on Bash and Ansible scripts to quickly generate testing and production-like environment.

During the Phase B technology pilot evaluation, there were no further requirements specified by the GÉANT team. The development of the use case led to a number of communications with the ONOS community and resolution of several relevant issues.

4 SDN-Based Circuit on Demand

4.1 Description

Circuit-oriented services are already a reality in several NRENs, including GÉANT, where AutoBAHN allows an end-user to request multi-domain services with a guaranteed bandwidth during a set period of time [AutoBAHN]. AutoBAHN provides the Bandwidth on Demand (BoD) service across multiple domains through the NSI-CS v2 protocol and custom made per-domain agents. It currently does not support SDN domains (OpenFlow-based or using any other protocol apart from the technologies for which specific proxies exist). The SDN-based circuit on demand use case of JRA1 relies on the DynPaC application to deliver SDN--based circuit services (currently based on OpenFlow but also potentially other soutbound protocols supported by an SDN controller). DynPaC is a solution for advance reservation of circuits with guaranteed capacity that allows provision of resilient VLAN-based L2 services taking into consideration bandwidth constraints,[1] while at the same time leveraging an OpenFlow-based interaction with the data plane [DynPaC]. Therefore, an on demand SDN-based circuit alleviates the need for custom domain proxies and is deployable in any network that supports standards-based protocols, such as OpenFlow.

With the introduction of DynPaC, support of the NSI-CS protocol is maintained. Support for OpenFlow domains is added by means of utilisation of the DynPaC framework as the Domain Manager for domains, as depicted in Figure 4.1.

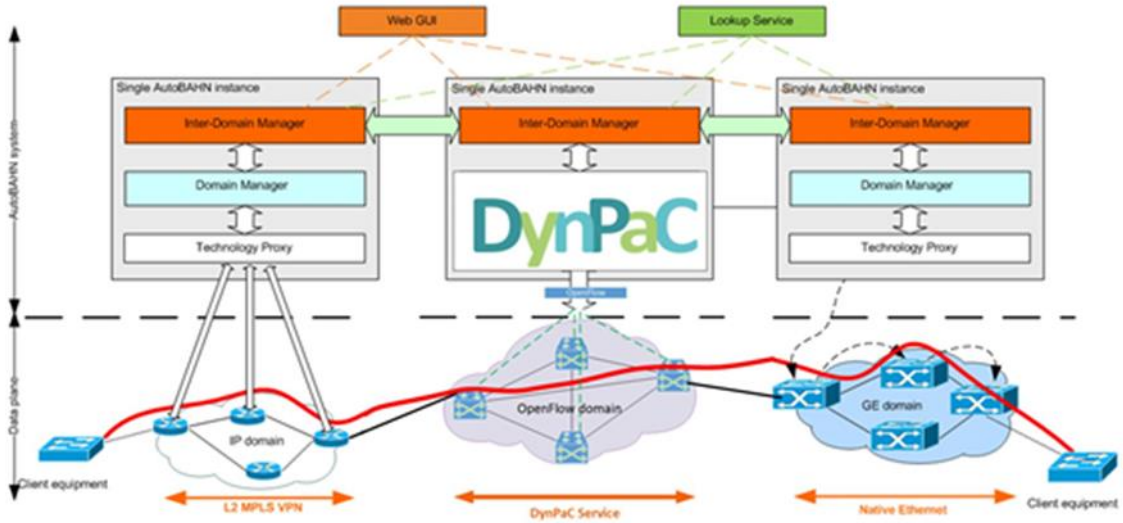

Figure 4.1: Using the DynPaC framework as a domain manager in the NSI architecture

[1] Similarly, the US NREN, ESnet, offers OSCARS to its users, which adopts the PCE-based architecture in order to compute the paths.

The following list summarises the features provided by the DynPaC framework, some of which are additional to what AutoBAHN/other existing circuit on demand solutions offer:

- Compute the best possible path between two end-points, taking into account:
  - Available bandwidth.
  - Available VLANs.
  - Minimum hop count.
  - Path stability.
- Optimise network resource utilisation.
  - Flow relocation mechanism.
- Resiliency.
- Rate limiting.
- Support for VLAN translation.
- Automated network programming.
- Automated network discovery.

In the context of the GN4-2 JRA1, DynPaC development focussed on implementing the missing required functionalities and meeting the required readiness level. For this reason, it was deemed necessary to redesign and refactor part of the application in order to:

- Expose the Service Manager component of DynPaC as an ONOS service.
- Improve the path computation element (PCE) and ultimately, make it stateless.
- Move resiliency handling to Service Manager.
- Improve topological changes event handling.

In the next sections, the setup and results from Phase B of the technology pilot evaluation that was conducted in April 2018 are presented.

4.2 Functional/Operational Overview

The technology pilot of the SDN-based circuit on demand use-case aimed to demonstrate the ability to rate-limit flows at the level requested and agreed, to verify the capability of the software to discover network topology automatically and react in a timely fashion to topology changes, make efficient decisions on admission requests, optimise the reallocation of flows, and act effectively on network disruptions for protected services.

A detailed list of desired requirements as defined during the use case specification was provided on the JRA1 wiki to internal participants.

4.3 Pilot Overview

Figure 4.2 shows the location of the Corsa switches in GÉANT POPs used for the SDN-based circuit-on-demand pilot.

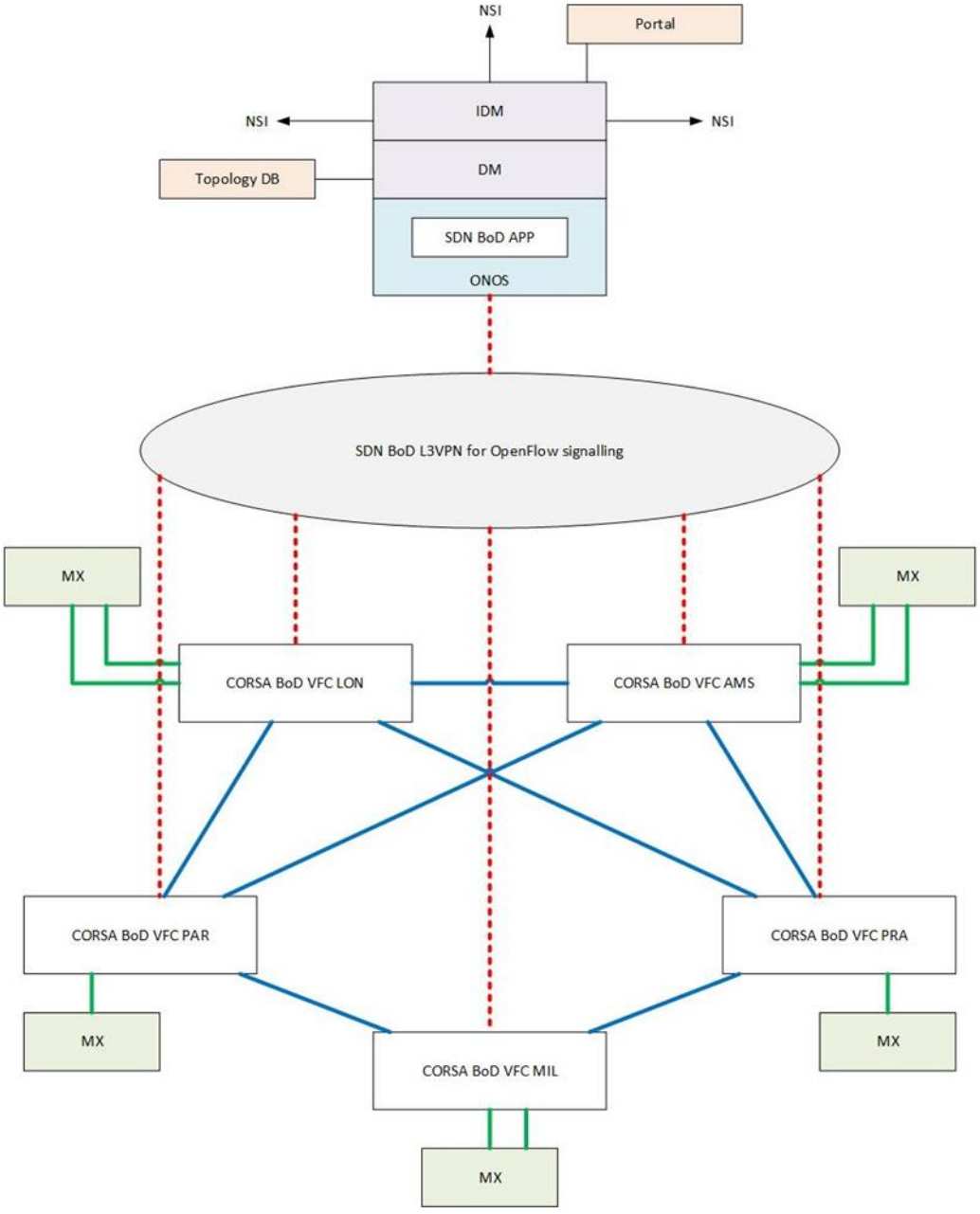

Figure 4.2: SDN-based circuit-on-demand pilot slice overview

In order for multi-domain aspects to be evaluated, the pilot was interconnected with the GÉANT Cambridge lab which was considered a separate domain, as shown in Figure 4.3.

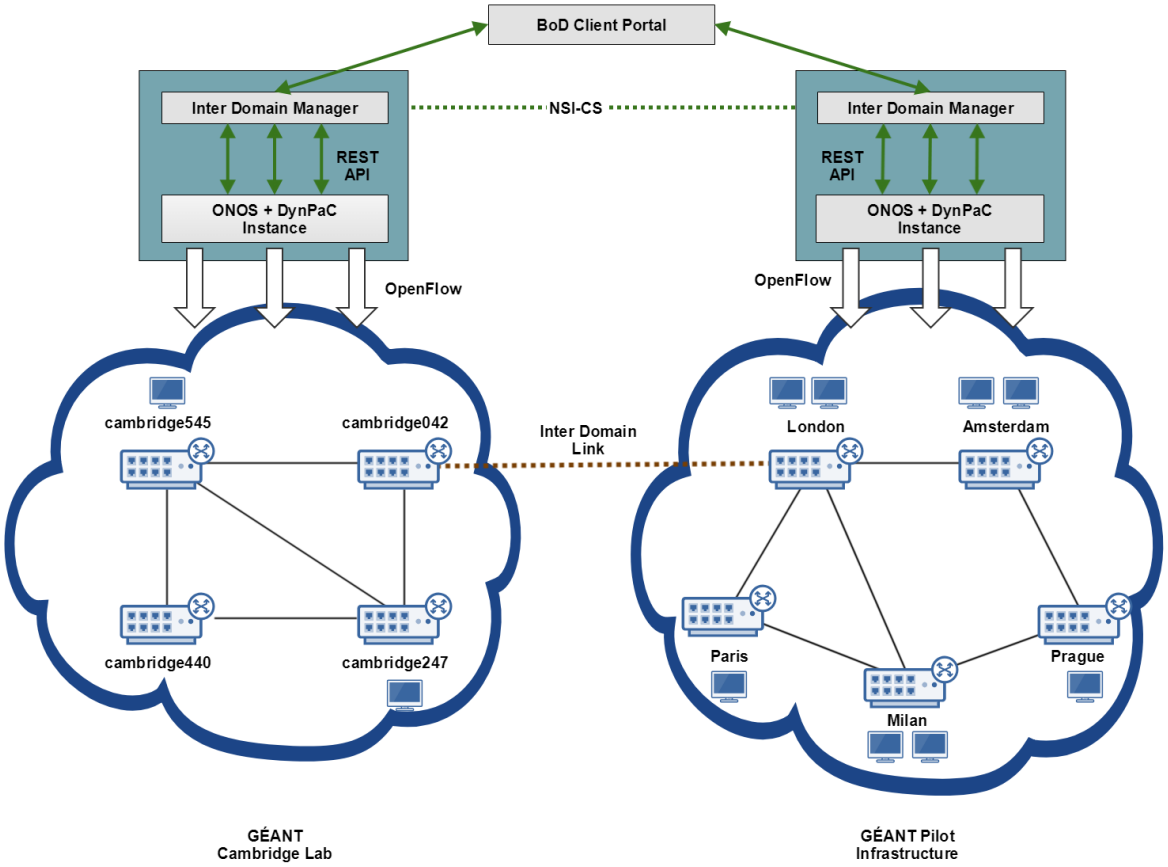

Figure 4.3: SDN-based circuit-on-demand pilot slice overview

The Phase B technology pilot focused on two fundamental aspects:

- The features provided by the DynPaC application to provide the circuits on demand.
- The behaviour of both the application and the ONOS controller to handle failures.

The DynPaC application works on a continuous refactoring process to improve the performance of the algorithms and add new functionalities, so its pilot evaluation has been divided into five different parts, each of which is associated to a different release of the DynPaC framework. The releases and the set of features included in each of them are listed below:

- DynPaC v.1.5 (Improved performance)
  - Bugs detected in previous demonstrators solved.
  - Basic VLAN translation
  - Meter handling.
  - Developed ONOS driver for HP3800 switches based on patches provided by Lancaster Uni; driver solved issues regarding filtering and order of OF instructions
  - Added basic handling of topology change update events
  - New network snapshots management algorithm.
  - Added support for parallel links
  - Verified and fixed handling of link up/down events

    - Changed path computation algorithm to ONOS implementation of K shortest paths
    - Asynchronous scheduling.
    - VLAN granular handling
    - VLAN efficient translation
    - VLAN configuration in config file by admin
    - REST API restructuring to support VLAN translation (IDM to request links and VLANs availability)
- DynPaC v.2.0 (Improved interface)
    - Add logs to facilitate troubleshooting
    - Improve GUI with new functionalities
    - Visualization of network snapshots
    - Path visualization
- DynPaC v.2.1 (Improved performance)
    - Restore the node aggregation policy in the network snapshot db
    - Run scaling tests
- DynPaC v.2.2 (Improved meter handling)
    - Select devices and create meters based on load
- DynPaC v.2.3 (Clustering and persistence support)
    - Refactorization to expose the service manager as a service.
    - Clustering support (either via ONOS clustering or external machine clustering).
    - Automate the driver discovery.

Regarding the suitability of the DynPaC framework to provide the guaranteed bandwidth, circuit-on-demand service, and taking into account topology discovery features and resiliency, the following list summarises the Phase B evaluation elements of the technology pilot. For this evaluation, the first stable release of the DynPaC application, v1.5, running as an application of the ONOS controller was used to check:

- If an unprotected circuit was correctly established, using the shortest path available to connect the specified source and destination nodes.

  This was also important to check in case of a link failure (the service is provided through the affected link), the backup path is installed using a disjoint path and that the primary path is deleted from the flow entries of the OpenFlow devices.

  If an unprotected circuit was correctly established, using the second-shortest path available to connect the specified source and destination nodes as a result of having any of the links of the first shortest path fully reserved.
- If a protected circuit was correctly established, using the shortest path available to connect the specified source and destination nodes. Notwithstanding, the backup path will not be the best disjoint path, but the second-best disjoint path, as a result of having any of the links in the best disjoint path fully reserved. In addition, the establishment of the backup path in case of a link failure and the proper programming of the OpenFlow devices needs to be tested.
- If the network is fully booked, a new service reservation is rejected.

- If peak bandwidth specified for an unprotected service is correctly enforced at the Corsa devices. In order to test that the bandwidth is correctly enforced, traffic needs to be injected and the meter tables of the Corsa devices and the received traffic need to be checked.
- If peak bandwidth specified for a protected service is correctly enforced at the Corsa devices. Once it is, check that the traffic is correctly shaped. The interface utilised by the primary path at the Corsa device is disabled to see if the devices are correctly updated to install the backup path, while the rate limiting is still enforced.
- If a protected service is still being provisioned after the failure of a node used by that service.
- If the network can be over-subscribed by the sequential request of multiple service reservations, and check if under such circumstances, the services experience any packet loss as a result of having the network operating to its fullest capacity.
- If the VLAN translation is correctly enforced for a service where different VLANs are requested for the ingress and the egress.
- If the circuit for a given service reservation is correctly established, and that the traffic is exchanged without any problem when Q-in-Q is used [Q-in-Q].

In addition, further evaluation analysed the impact of failures on the control channel or at the controller side. This tested the behaviour of:

- The switches when the connection between all the switches in the network and the ONOS controller, and therefore with the DynPaC application, fails. In this regard, the behaviour at both the controller side and switch side was tested, to see the logs that are generated and if the flow entries associated with the service reservation are still active or not.
- The switches when the connection between all the switches in the network and the ONOS controller, and therefore with the DynPaC application, is restored after a disconnection. In this regard, the logs generated at the controller side were tested, as were the flow entries of the devices.
- A given switch when the connection with the ONOS controller, and therefore with the DynPaC application, fails.
- A given switch when the connection with the ONOS controller, and therefore with the DynPaC application, is restored.

Furthermore, given the fact that the circuits will be provided in a multi-domain fashion, the following evaluation was also conducted:

- Check if DynPaC is able to retrieve the topology of the network correctly, perform the necessary abstractions to ease the path computation and ensure the STPs that will be advertised to the NSA are correctly identified.
- Check if DynPaC correctly advertises the topology information related to the domain through the REST interface to the NSA.
- Check if a multi-domain circuit is established correctly when requested from the GÉANT's Bandwidth-on-Demand portal.
- Check if a multi-domain circuit is removed correctly when it is requested from the GÉANT Bandwidth-on-Demand portal.

In addition, taking into account the relevance of monitoring the status of the circuits and the network elements, a set of tests related to operations and management were also conducted:

- Check that it is possible to retrieve the number of packets transmitted on a given switch on a per-port basis.
- Check that it is possible to retrieve the number of packets transmitted on a given switch on a per-flow basis.
- Check if it is possible to retrieve the list of services currently being provisioned in the network.

4.4 SDN-Based Circuit on Demand Pilot - Detailed Results

Detailed results obtained from SDN-Based Circuit on Demand pilot Phase B are detailed in the SDN-based Circuit on Demand pilot results appendix located on the GÉANT project Wiki.

Table 4.1 below summarizes the results of the tests described in Section Pilot Overview4.3.

ID	Requirement	Result
DP1	Unprotected circuit	Success
DP2	Protected circuit	Success
DP3	Unprotected circuit with congested shortest path	Success
DP4	Protected circuit with congested second path	Success
DP5	Fully booked resources	Success
DP6	Ingress rate policing	Failure (partial)
DP7	Protection after link failure	Failure (partial)
DP9	Overbooking	Success
DP10	VLAN translation	Failure (partial)
T1	View topology database	Failure (Partial)
T3	Add new trunk to topology database	Success
T5	Add client to topology database	Success
MD1	Topology database export	Success
MD2	Multi-domain circuit creation	Success
MD3	Remote multi-domain circuit creation request	Skipped
OAMP1	Packet count monitoring	Success

ID	Requirement	Result
OAMP2	Packet statistics	Success
OAMP3	Retrieve details of individual service	Success
OAMP4	Retrieve list of services	Success

Table 4.1 SDN-Based Circuit on Demand test results summary

The majority of the tests were successful. Additional clarifications/details can be found below for specific tests:

- DP6 - DP7 are related to traffic rate-limiting via appropriately installed meters. During the actual testing, meters exhibited: (a) inability to appropriately meter the incoming traffic (DP6) and (b) inconsistencies related to the actual amount of allowed traffic (DP7). Note that despite the failed tests, the appropriate flows and meters were installed in the CORSA boxes. Moreover, the same testing scenario was revisited in GÉANT's Cambridge Lab whereby tests were successful and the traffic was consistently rate-limited, as configured. Although there is no definite conclusion as to the exact reasons, this is most probably attributed to the Software Version (2.1.5) installed in the Pilot Switches. The exact problem reported about this SW version is that meters exhibit a considerable amount of delay time (varying) for actually rate-limiting the traffic. The equivalent SW version for the Cambridge Lab switches is 3.0.2.

- DP10 - VLAN translation (Ingress VLAN <-> Egress VLAN). The tests related to the VLAN translation did not pass in the Pilot Infrastructure, however it is worth stating that: (a) the flows and meters were examined and appear to be correctly installed in the CORSA switches (visible both from ONOS and CORSA CLI) and (b) VLAN translation works without any problems in the Cambridge Lab. Regarding the Control Plane both setups share the same SW versions for ONOS and DynPaC. It is important to note that the two setups differ in terms of the data plane configuration. Specifically, regarding the Pilot environment the currently configured substrate between the test VMs is quite complex and there are no definite conclusions as to the exact cause(s) of the issue. Also note that, the test denoted MD2, which refers to an inter-domain circuit, is successful. The inter-domain circuit is comprised of 2 separate intra-domain circuits (both applying VLAN translation logic) one for each domain (Cambridge and Pilot).

- T1 - View Topology Database. The reason this was marked as a "Failure" is the inconsistency in the test description. The designated CLI command provided the inter-switch links (trunks) whereas the REST API provided the available edge ports and allowed VLANs.

- MD3 - Remote multi-domain circuit creation request - was skipped because in the current multi-domain setup it is not feasible to test and it ends up being a repeat of MD2.

- During the Phase B of pilot there was also general feedback on suggested software improvements, the most important was namely: (a) It is desirable to have human-friendly labels for the switches (loading an appropriate JSON through ONOS resulted in instabilities),

(b) removal of a service after a protected link failure and reallocation was not handled properly resulting in a software exception.

4.5 Findings and Future Work

The DynPaC framework has undergone a deep refactoring process to improve the algorithms used for path computation, advance reservations, topology handling and resilience. The necessity for such refactoring was identified during the previous phase of the project in order to increase the scalability of the solution. As such, several improvements have been applied to the application, while others are still pending.

Overall, the resilience and topology handling of the framework have been improved. In the previous version of DynPaC, the topology was retrieved at the activation of the application and remained unchanged. Only the failures of nodes already retrieved in that initial topology were handled, which imposed some limitations to the solution. The new version of the application is able to detect the addition of new nodes to the network, as well as the removal of nodes, and react to the failures by installing the backup paths and modifying future reservations, taking network changes into account.

The advance reservation system has been improved by the introduction of a new database of network snapshots (recording the use of network resources at different points in time). In the future, this network snapshots database will allow to aggregate multiple network snapshots hierarchically, in order to perform the admission control process in a much more efficient way, avoiding the linear behaviour of the current version.

In addition, there are still many pending features to be included in the DynPaC framework, mostly related to the resilience capabilities of the application. Future work (at least until the end of the GN4-2 project) can be summarised, as follows:

- Add clustering support to make the circuit on demand provisioning resilient to ONOS instance failures.
- Add a mechanism to retrieve the list of service reservations when all the ONOS instances fail. This is a very important feature, since the total disconnection of the switches and the ONOS controller results in the removal of all the flow entries installed in the network devices once the connection with the ONOS controller is back online. As such, and given the fact that all states of the service reservations are kept within the DynPaC application and not on an external element, all the information related to the reservations will be lost, and the services that were already running in the network would not be appropriately restored. As such, an external database to store the reservations is necessary, as well as the means to restore the switches to their previous state.

In conclusion, the existing software can be considered close to production readiness for the use cases and projected deployment. This conclusion stems not only from the pilot results detailed in this section, but also from the usage and testing of the software in the Janet and EHU testbeds detailed in Appendix A and B, respectively.

5 SDX L3 Status

5.1 Description

Functionality for the SDX L3 use case is based on an ONOS northbound application developed by GN4-1 and GN4-2 projects, namely the SDX L3 application. This application has been designed as a variant of ONOS SDN-IP [SDN-IP], a native ONOS application distributed with ONOS. SDX L3 supports all the features that SDN-IP provides, which can be summarised as follows:

- Interconnection of the SDN network with external ASes using the BGP protocol.
- Addition of the ability to the SDN network to act as a routable network, that can also transit traffic between external ASes.
- Moving the routing logic from the SDN network routers to the SDN fabric switches.

Consequently, the SDN-IP application allows the whole SDN network to act as a router by translating the routing information into appropriate rules for the programmable switches. The SDN network routers are still necessary, but their responsibilities are restricted to the BGP message exchange, thus router offloading is achieved and the routing performance of the SDN network is increased with significant cost reduction.

SDX L3 is targeted at control of an SDX fabric, therefore SDX L3 can support the case where all routers of the peering networks have interfaces in the IXP subnet. Thus, peering networks are able to directly exchange IP traffic without needing an additional routing hop inside the SDN network. SDX L3 can be combined with a BGP route server, which is a common type of router in current IXPs. Therefore, SDX L3 remains compatible with legacy deployments, and also takes advantage of router server features for simple IXP configuration and router efficiency. Figure 5.1 provides an overview of an IXP using ONOS with SDX L3 application. For simplicity, a single instance deployment instead of an entire cluster is depicted.

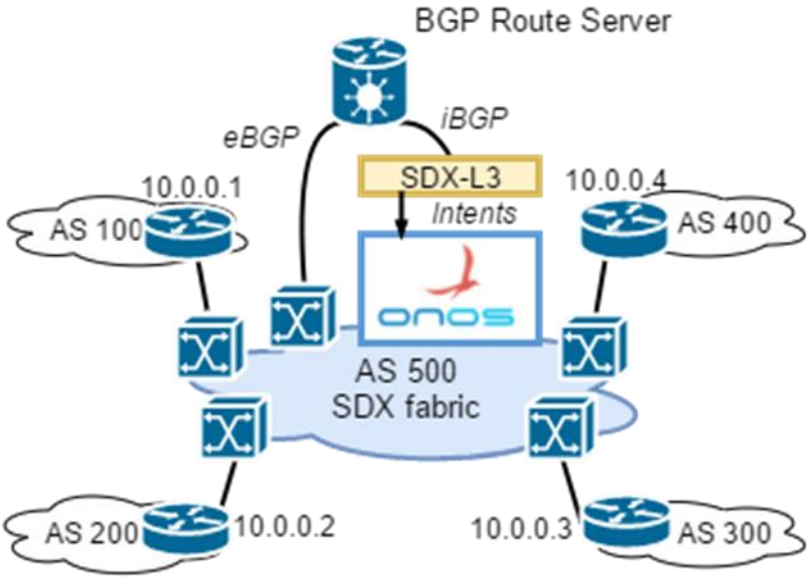

Figure 5.1: Sample IXP with a single ONOS instance with SDX L3 combined with a BGP Route Server

The GN4-2 project work on SDX L3 has focused on the following:

- Adaptation of the SDX L3 application to the ONOS evolution. Since ONOS is a very active project, there are frequent updates at the APIs as well as introduction of new features. SDX L3 software has been regularly adapted to stay compatible with ONOS recent releases and take advantage of the new or improved ONOS features. In order to make the software ready for deployment, the use case also participated in the ONOS community "Deployment Brigade" [ONOS_BRIGADE] activities.
- Integration of the ONOS SDX L3 specific 'corsa-l3-overlay' driver developed by Corsa for Corsa DP2000 series switches.
- Extensive evaluation of SDX L3, the Corsa 'corsa-l3-overlay' and ONOS in the laboratory testbed setup on the GÉANT premises in Cambridge UK. This is the most important step, and its results form a tollgate, since they validate the software and evaluate its state prior to pilot phase. Such evaluation is important, especially in terms of system scalability, which is a major concern for L3 production networks. SDX L3 testing examines whether the needs of GÉANT and NRENs can be adequately met.

5.2 Current Status of SDX L3

The current status in Phase B remains as summarized in the SDXL3 traceability matrix [SDX-L3] in the form of lab testing, as this use case has not matured enough to be considered suitable for a technology pilot.

A simplified version, with the status at the time of writing, is provided in Table 5.1. The information presented here is unchanged from Phase A.

Functionality group	Requirement / Test title	Result	Remarks
L3 Functionality	IPv4 Support - BGP Transport Between BGP Peers	Success	
	IPv6 Support - BGP Transport Between BGP Peers	Fail	
	VLAN Support - BGP Transport and IP Routing for Different VLAN Setups	Fail	
Redundancy, High Availability and Failures	Network Status After a Complete CP Failure (Single Controller Scenario Only)	Fail	
	SDN Control Plane Failure Recovery (Single Controller Scenario Only)	Pass	Partial results
	Resiliency Against Link Failures	n/a	
BGP Functionality	Add, Remove or Shutdown BGP Peerings Without Impact on Other Peerings	Success	
	Possibility of IBGP and EBGP Peering	Pass	Partial results

Functionality group	Requirement / Test title	Result	Remarks
	MD5 Support	Success	
Switch-Related	LACP Support for Bundling the Links Between Switches	n/a	Test was not defined
Monitoring and Operations	Correctness and Intuition of ONOS SDX-L3 Commands	Success	
	Export of the Flow Statistics From the Controller	Fail	
	Coexistence with Other Applications	n/a	Prerequisite tests were failed
Scaling	IPv4/IPv6 Prefixes Supported in the order of hundreds of thousands/1 million	Fail	

Table 5.1: SDX-L3 test results summary

Not all of the tests have been successful and it has not been possible to perform all tests, due to limitations in the given test setup. The main issues related to the failed requirements are IPv6 testing (Test IPv6 support - BGP Transport Between BGP peers) and mixed tagged/non-tagged traffic (VLAN Support - BGP Transport and IP Routing for Different VLAN Setups test) for the general Layer 3 functionality testing.

By the time most of the tests were completed, a new driver for ONOS was provided by Corsa and selected tests had to be performed with this new driver. The driver 'corsa-l3-overlay' has been designed to efficiently support SDX L3 functionality on Corsa DP2000 series switches. The following tests were successfully performed with the new Corsa driver:

- IPv4 Support – BGP Transport Between BGP Peers.
- Export of the Flow Statistics from the Controller.
- Number of IPv4/IPv6 Prefixes Supported.

Regarding redundancy and high-availability features:

- The Network Status After a Complete CP Failure (Single Controller Scenario Only) test failed, but only in the case where the route server feature was activated. The test was successful without that feature.
- Due to the fail result of test IPv6 Support - BGP Transport Between BGP Peers, the IPv6 protocol was not tested in SDN Control Plane Failure Recovery (Single Controller Scenario Only) or SDN Control Plane Failure Recovery (Single Controller Scenario Only).
- The Resiliency Against Link Failures test was not performed, since the lab topology does not allow for it.
- Regarding the specific BGP functionality testing, all tests were successful, however the following should be noted:

    - The Possibility of iBGP and eBGP Peering test was not fully performed, due to missing topological elements.
- For the same reasons, the LACP Support for Bundling the Links Between Switches test was not performed.

Regarding monitoring and operations testing:

- The Coexistence with Other Applications test has not been performed, since the previous generic tests were not a complete success.
- The scalability test, Number of IPv4/IPv6 Prefixes Supported failed, indicating poor performance beyond thousands of BGP routes (max 6000 intents were installed). The results were even worse after the new driver (Corsa-overlay) was used – as there were only just over one thousand intents installed on the Corsa switches.

During the testing process, instabilities with the setup and the base ONOS controller have seriously affected the execution of the tests. The main problem is related to the ONOS in the lab setup and appears to have a number of causes:

- When ONOS process is restarted with "service onos restart" command, the Corsa driver often does not load properly, so it has to be reloaded after every restart.
- It appears crucial to isolate ONOS from SDN switches and to deactivate the SDX-L3 application prior to following the process.

It is not clear whether some of the results (e.g. in the Network Status After a Complete CP Failure test) have been affected by this.

It should be noted that the Corsa driver for the multi-table pipeline is considered to be far from finished, as scalability issues within ONOS have been identified. Thus testing with the driver was restricted to understanding the limitations ONOS presented and considering options for improvement. The main outstanding driver issues include the following:

- Each multipoint to single point (MP2SP) intent is completely independent, which causes each advertised prefix to allocate a separate group. This causes a scalability issue due to the inefficient management of available groups. (The pipeline supports 1022 groups which is in fact quite a large number of next hops for any router.) According to ONOS developers, this is the current state-of-the-art in the intents framework, and significant work would be needed to overcome this. One option for the driver would be to try to flatten common next hops, however, this is considered a heavy burden and leaks application knowledge into the driver. In order to get to a significant scaling capability this approach may be required, although it has some architectural drawbacks without clear resolution. Initial work on the Corsa driver used a single, advertised prefix per router.
- As a side-effect of the above, any common flows that are generated as a result of installing flow objectives need to be reference counted by the driver in order to avoid the situation where a withdrawal of an intent affects a flow used by other users. In a single-table design, this is avoidable due to sufficient selector differences that make each flow unique. However, in a multi-table design, common flows may fall out (e.g. coming in on port x in table n should transition to table y). The driver is currently lacking this reference counting logic.
- The objectives coming out of a MP2SP intent cause a great number of 'ripple effects'. Whereas ideally, the next hop path from ingress switches to egress would be tacked up once, and

adding/removing prefixes should merely change the FIBs on the ingress routers. Much churn triggered by flow changes.

- Withdrawal of intents is problematic due to unresolved sequencing/race issues where next/forward objectives are occasionally not sent in the right order.
- Flow objective store clean-up is not fully implemented.
- The driver is taking advantage of device-specific group IDs. With Corsa using group range 0-1021, it is important to ensure that groups are mapped into this range. As the group IDs in ONOS are by default global, it is possible to cross outside the valid group range.

The above issues are known to Corsa, which has undertaken the development of the ONOS driver.

5.3 Future Work

The laboratory evaluation of the SDX L3 use case showed that there are major issues regarding system scalability compared to a realistic (set) number of routing prefixes. Known ONOS platform problems in the version tested include:

- Problematic system recovery after failures, to an extent that 'system-clean installation' is required to achieve proper operation.
- Immature driver subsystem makes driver configuration laborious and error-prone.

However, the most important SDX L3 deficiency is its poor scalability, which cannot be improved by the deployment of the specialised Corsa SDX L3 driver due to the inefficient management by the controller. This makes the system scale far less than the desired levels (magnitude of thousands against hundreds of thousands or millions of IPv4/IPv6 prefixes supported). A flow rule compression is imperative for the controller to adequately support the SDX L3 use case. Since ONOS is a general purpose SDN controller, it is difficult to achieve the desired flow rule compression, therefore a custom ONOS intent framework that is tailored for SDX L3 could be a possible solution to this direction.

One possible future direction of the use case is a more sophisticated route selection mechanism that does not rely only on BGP primitives. When faced with the problem of the congested edge links, it is important for the backbone network to make a decision that takes into account the capacity of the links and the packet loss. To this direction, there are several different sources of information that can be used by the control plane exclusively or collectively, such as full visibility of the peer prefixes via BMP protocol, flow monitoring on interfaces via NetFlow protocol, as well as link utilisation via SNMP.

In addition, an investigation of how P4's support for hardware acceleration could also improve scalability for SDX L3 cases is also possible [P4].

Clearly, the existing issues and limitations prevent the SDX L3 solution from being deployed in a pilot environment.

6 Transport SDN Status

The Software Defined Network (SDN) paradigm could be used to dynamically provision and manage the capacity of optical connections in transport networks. Considering that the current GÉANT optical network is based on Infinera's equipment, GÉANT decided to test Infinera's implementation of the Open Transport Switch (OTS) as part of its Phase A piloting activities.

The Infinera OTS is a lightweight virtual transport switch that provides an SDN interface to the DTN-X equipment. The OTS adds transport extensions to the SDN interface that can be used to control OTN transport circuits. However, the interface is not OpenFlow-compliant. Communication between OTSv and ONOS is through a JSON-based proprietary REST API by Infinera. OTS could also be used to offer deterministic Layer 2/circuit services over the optical layer.

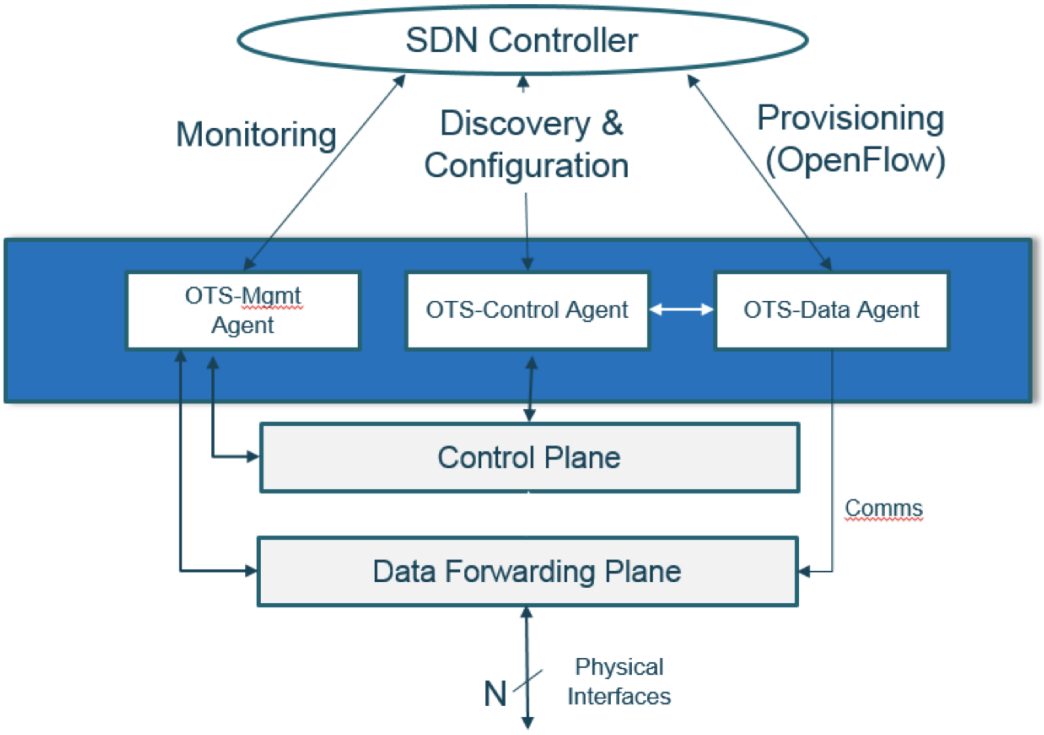

Figure 6.1: SDN architecture for the Infinera OTS

Table 6.1 summarises the major requirements defined for this pilot.

Requirement	Description
EVPL on demand	The Infinera equipment used in this trial must support Ethernet Virtual Private Line (EVPL) services (referred as L1-EVPL).
Infinera OTS enabled	The Infinera equipment used in this trial must be at software version 11.03.
DTN-X in Cambridge lab with 2 PXM units	The Infinera equipment used in this trial will be installed in the Cambridge lab. This requires 2 PXM units and two OTM2 units.

Requirement	Description
OTS compatible	The controller must be compatible with the Infinera OTS.
REST support	The controller used must support the REST interface used by the OTS.

Table 6.1: Transport SDN requirements

For consistency with the other use cases, and to allow future integrations of the Transport SDN use case with upper layer services (e.g. to exploit multi domain scenarios, such as L1, L2 and L3, under the same control), we have chosen to leverage on the Open Networking Operating System (ONOS).

6.1 Current Status of Transport SDN

While ONOS already offers some abstractions to manage optical devices, it completely lacked two main requirements of our use case (Row 4 and 5 in the requirements listed in Table 6.1):

- ONOS is intended to act as single control plane software and is designed to be directly connected to the devices (both optical and IP) that ONOS manages. However, Infinera does not allow such direct interaction with the physical devices, and offers a central management software (OTS) that is in charge of the configuration. This is common behaviour for transport vendors that usually prefer to hide the complexity of the optical feasibility estimation and to expose a simplified northbound interface to collect information and require for services. To overcome this limitation, in collaboration with the ONOS engineers, we had to design and develop a new ONOS Adapter as an extension of the existing REST provider, to support multiple devices under the same REST connection. This modification required the improvement of several ONOS modules to offer this new functionality, while being still compliant with the standard behaviour. The developed code has been included in the ONOS codebase, starting from ONOS 1.9 release.
- The specific REST API and workflow implemented by the Infinera OTS has to be supported by the SDN controller. The ONOS southbound interface is designed to be modular and easily extensible. Following this approach, the REST adapter, composed of a provider and a protocol module, is generally enough to offer an abstraction for all the use cases and solutions that leverage the REST protocol to communicate with the Infinera DTN-X devices. However, to support vendor-specific extensions and interfaces, ONOS allows definition of drivers that are associated with specific equipment and implement the detailed calls and workflows supported by the devices. Following this approach, in Phase A we have designed, developed and tested a new driver to manage Infinera DTN-X devices through the northbound REST API exposed by OTS. This interface is covered by an NDA, so the code is securely stored in the GÉANT repository and is not shared with the ONOS community.

After the development of the ONOS driver for the support of Infinera OTSv node and the appropriate REST protocol extensions for the communication at the southbound interface (SBI), a major challenge since the beginning of the GN4-2 project has been the complete integration of OTS with ONOS core, i.e. also with the ONOS Intent Framework. This integration would allow any ONOS northbound application to use the ONOS' intent abstractions to control the optical transport network devices and

would pave the road for multilayer support, that would allow combination of GN4-2 L2, SDX-L2 and SDN circuit on demand use cases to be combined with SDN transport under a common control plane.

The major challenge of this complete integration is the fact that the ONOS interface assumes that it communicates directly with the switching devices at its southbound, as previously mentioned. However, OTS adds another level of abstraction and acts as a "proxy" or "child-controller" of the optical transport network. As a solution, the "Domain Intent" concept emerged via the collaboration of GN4-2 engineers, together with people from other R&E projects, namely "ACINO" [ACINO] and "ICONA" [ICONA], as well as ON.Lab experts. The Domain Intent solution addresses the case of a child-controller, sitting at the ONOS southbound, which is responsible for an external administrative domain, thus forming a hierarchy of controllers. When a request for policy application, e.g., for some connectivity at a given rate, concerns devices that belong to an external administrative domain, ONOS is able to identify this particularity and requests the installation of any policies, not directly, but via the child-controller. Domain intent is the abstraction that is used for the communication of the policies of an external administrative domain. The proposed implementation of the "Domain Intent" solution was approved by ON.Lab and is included in the ONOS codebase starting from ONOS 1.10 release.

As the capabilities of the transport layer of GÉANT do not include full-scale development of the OTS in production, the delivery of the specific use case in production has been ruled out and therefore Phase B piloting has not been conducted.

7 Working with Corsa

Corsa Technology solutions have been initially tested in the JRA1 lab environment, starting with the DP6400 switching platform. During the course of this testing, the task has closely collaborated with the vendor in order to obtain support for the multi-table pipeline driver needed by the SDN controller in order to install flows.

Subsequently, Corsa made its newer platform of DP2000 switches available, which was obtained by GÉANT both for the lab and the pilot environments. A major feature of the DP2000 platform is the support for Virtual Forwarding Contexts (VFCs), and the resulting capability to present various types of virtual switches to the SDN controller. These include, among others, a generic, single table OpenFlow VFC type and a multi-table Layer 3 VPN VFC type designed for IP Routing. The former VFC type was selected for SDX L2 and SDN circuit on demand use cases, while the latter was selected for SDX L3/SDN-IP, due to its focus on scalable IP Routing.

During the development and experimentation with the new switches, JRA1 was in constant communication with Corsa engineers that provided the support and development assistance into getting the software operational with the hardware platform. GÉANT and JRA1, on the other hand, provided feedback to Corsa for bug fixes and feature suggestions, as realised during the testing.

The most important issues that had to be resolved in collaboration with Corsa over the course of the project and pilot preparations included:

- Support for metering at VFC layer of Corsa DP2000 in order for meters to be visible by the SDN controller when controlling a virtual switch.
- Implementation of an ONOS driver for the multi-table pipeline of the L3VPN-type VFC instantiated within Corsa DP2000.
- Bug fixes observed during testing of the use cases over the Corsa equipment.

All of the above issues were handled and ultimately resolved in coordination with Corsa.

Furthermore, a number of issues were investigated in collaboration with Corsa, and were deemed to be issues related to software layers (ONOS controller and applications), thus resolved after proper controller adjustment. The most important issues were:

- Support of a group table for VPLS multi-point operations instead of multiple-output action generation.
- Filtering of clear deferred actions in the context of a Corsa-specific extension of ONOS default single table pipeline driver.

8 Working with Hewlett-Packard Hardware: Janet and EHU Testbeds

The HP3800 switch provides OF 1.3 support in the hybrid mode. The model supports all the OF functionality required by DynPaC, including:

- Forwarding rules based on VLAN IDs and ingress port numbers.
- VLAN translation.
- Drop meters.

HP3800 switches are used in the Janet testbed, and also in the EHU deployment (HP Aruba 3810). It was decided that it would be useful to test DynPaC on HP3800 switches to see how the software works on the switches of a popular vendor, and whether it has any vendor-specific features. Ideally, it shouldn't have any, as the corresponding vendor driver should mask them.

Another reason to test DynPaC on the Janet testbed was for its suitability along with the main GÉANT pilot infrastructure deployment in the multi-domain scenarios.

More details of testing DynPaC in the JANET lab are provided in Appendix A and in the EHU experimental facility in Appendix B.

8.1 Findings

The problems discovered during DynPaC Phase A testing in Janet SDN testbed have been:

- ONOS (up to version 1.10) does not support HP3800 switches: it does not have a specific driver for HP3800 switches and the ONOS default driver uses table 0 as default table instead of table 100 used by HP3800.
- DynPaC generated flow rules using an action order unsupported by HP3800: the first action uses the output port number, the second one - the VLAN ID, but the order should be the opposite.
- ONOS can't detect links using BDDP when switch ports are not assigned to the first VLAN ID of a OF instance (VLAN 150 in the testbed configuration).

Appendix A provides further insight to how these issues were addressed by the JRA1 team.

All the works inside the Janet testbed created a great opportunity for ONOS and DynPaC software to be tested in another close-to-production environment. As a result of this work, the driver for the HP3800 was developed. At the end of research and development activities, the new software has been tested in the ONOS 1.10.0-SNAPSHOT and ONOS 1.11.0-SNAPSHOT environments, with the two HP3800 switches. After testing in the Janet lab testbed, the driver code was submitted for the ONOS code review, as it was important to share the outcome with the ONOS community. New commits were pushed to the DynPaC repository for VLAN range configuration and priority value configuration for flows using DynPaC.

In the EHU testbed, a bug that applies to HP Aruba 3810M switches for several specific firmware versions was identified, reported to and acknowledged by HP. The bug appears when a meter is inserted by the ONOS controller in one switch. The meter is inserted correctly, but the reply of the switch doesn't reflect it, and shows that there are no meters inserted. The controller tries to insert it again, and since the meter is already inserted, the switch sends an error message telling that the meter already exists. The temporary rectification, until the vendor fixes the bug, has been to find an unaffected firmware version and work with it without executing firmware updates. A detailed presentation of the EHU testbed is provided in Appendix B.

9 Working with ONOS

GÉANT joined ONOS core members and other affiliated organisations in a working group called Deployment Brigade [BRIGADE]. Brigades are teams focused on developing specific features to ship in the upcoming versions of ONOS. Once formed, anyone else in the community with interest in such features is able to join the group and contribute towards it. The Deployment Brigade was the first attempt to use the model in the ONOS community, which turned out to be an effective way to recruit community members and ship new features.

The goal of the Brigade is to build a software stack running on top of ONOS, able to facilitate the adoption of the platform in field trials and production deployments by RENs and operators. Basic requirements include: providing Layer 2 and Layer 3 functionality to network users, convergence of packet and optical resources, and compatibility with major standards. After only a few months of work, the Brigade released non-trivial software components, already part of the SDX L2 Pilot within GÉANT. Figure 9.1 shows the converging stack the brigade members agreed to work on. Using the same applications allows the whole group to benefit from better test coverage, to obtain more significant feedback from a larger community and receive better support from the core developers while still allowing for minor deployment-specific customisations.

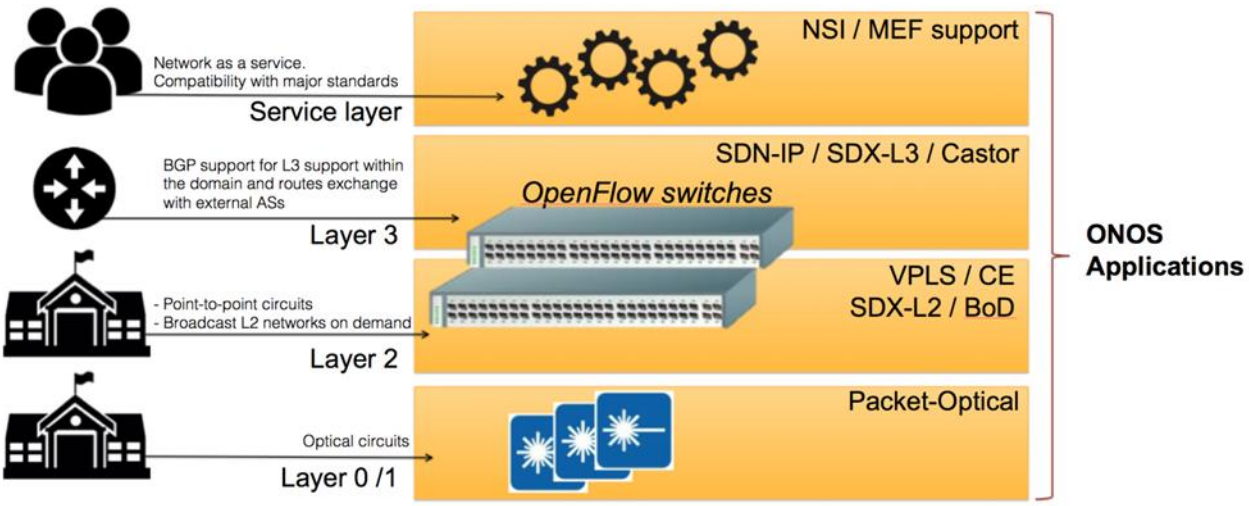

Figure 9.1: Software stack under development by the Deployment Brigade

The stack that the group decided to share and improve includes applications to help operators manage different network layers. Specifically, VPLS [VPLS] is used to create on-demand Layer 2 broadcast networks, Carrier Ethernet [CE] supports MEF standards for the creation of Layer 2 services such as E-Lines (point-to-point circuits) and E-LANs (multi-point broadcast networks), SDN-IP [SDN-IP] transforms SDN networks into IP transit networks, translating BGP messages exchanged with external ASes to OpenFlow entries on the switches inside the domain, and packet-optical [PO] is able to manage the packet and the optical layers, both during normal operations and after network failures.

The activities of the Deployment Brigade include the design, the development of new software components, and their integration; Quality Assurance and deployment activities. Members usually take part in different sub-projects and activities, depending on their background and interests. The

Deployment Brigade has produced a two-fold contribution: first, it worked with GÉANT to evaluate and take advantage of new SDN solutions, thereby encouraging a wider community to be part of this work and improve its chances for success. Second, it gave back to the ONOS community with significant artefacts (including new applications, features, improvements and bug-fixes), made available in the ONOS repository and to other network operators.

The Brigade initiative has been so successful that the ONOS project expanded the number of active Brigades in 2017. The Brigades initiative also created the opportunity for GÉANT team member internships at ON.Lab, enhancing the communication between the team members and shortening the learning curve of the ONOS architecture. This way of working allows GN4-2 project teams to produce outcomes directly to the open-source code base and make the project visible to a part of the industry that is starting to see SDN as something more tangible. Also, participating in the Brigades has produced an unexpected benefit to the team in the form of advice and patches.

In the spirit of open source, all the developed applications have been made available to the broader community. The work of the Brigade and the results of the tests have been showcased and demonstrated at different international meetings and conferences. The Brigade has been extending the ONOS intent framework to introduce the notions of queues, meters and bandwidth, and is evaluating how the actual applications can run over the framework once all the functionalities are in place. As opposed to working as stand-alone applications, both DynPaC and CE will be integrated with the stack, bringing compatibility with the NSI and MEF standards while reusing the primitives offered by SDX-L2 and VPLS. SDX-L3 and SDN-IP will be the first integrated and then merged with the rest of the stack; tested and deployed. Finally, additional tests will be conducted before moving the platform to production, in order to ensure the robustness of the solution before it will be offered as an official service to the broader community.

The ONOS driver developed for Janet HP3800 switches within BoD use case was contributed to ONOS codebase, as mentioned above, and has been merged to ONOS master branch (1.11.0-SNAPSHOT) and ONOS 1.10 support branch (1.10.3-SNAPSHOT).

10 OF Conformance Scripts

In order to verify the progress of the Corsa firmware and support piloting activities, a suite of tests has been developed. The tests are realised using OFTest [OFTest], a community project started by Big Switch Networks [BIG_SWITCH]. It is a python based framework originally developed to test the compliance of the switches with the OpenFlow (OF) protocol. Ideally, these tests should be run at each release of Corsa firmware before proceeding with upgrades to the Corsa devices in the lab and in the pilot infrastructure. The objective is to identify in time regressions or eventual problems and avoid unnecessary downgrades.

The test library has a minimum set of requirements, it needs a host or a virtual machine equipped with three interfaces (two interfaces are connected with the data-plane in a closed loop and one is used for the control-plane). The switch to be tested needs to have two ports and a management interface. If the VM can be equipped with more than two interfaces, it is possible to create several VFCs on the switch and test different pipelines/configurations in parallel without the need to re-configure everything. Figure 10.1 shows the setup for executing the GÉANT OF conformance tests considering a multi-VFC scenario.

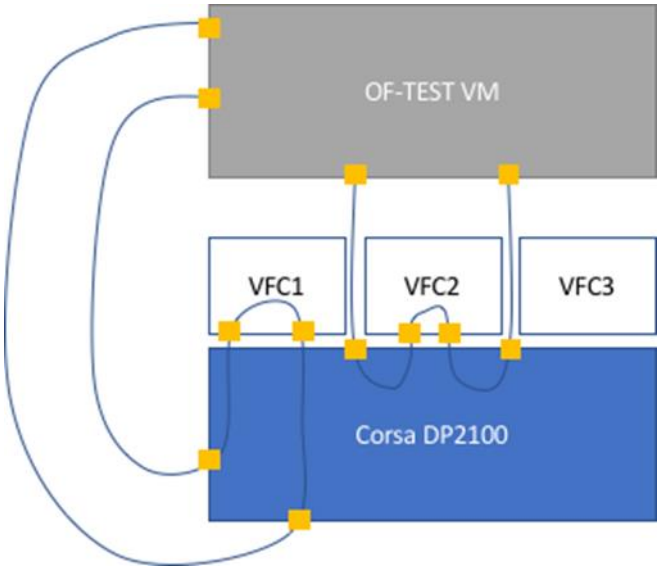

Figure 10.1: OF conformance setup

The workflow of tests is very simple and straightforward; it requires:

- Programming the switch according to the functionality to be tested.
- Sending a packet in the data-plane, using one of the available interfaces.
- Verifying that the packet received on the remaining interface matches the expected packet.

In order to facilitate test development, a virtual environment has been developed which does not require the presence of a real hardware switch as it emulates the Corsa DP2100 devices using a software switch.

As part of the solution, several tests have been developed. The tests are organised into four main categories: i) Flows, ii) Bugs, iii) Feature requests, and iv) Groups. *Flows* includes all procedures meant to verify the basic functionality supporting JRA1 use cases, such as MAC address forwarding, VLAN switching, and so on. *Groups* has been developed to assess the capabilities of Corsa devices of handling more advanced functionalities like OF Groups. The Bug group contains all the tests developed to verify the progress of the Corsa software against the identified bugs and will be useful to try out the future firmware updates. An example of this it has been the *Clear Deferred* action, which is currently supported by Corsa switches, but it had been unsupported for several releases of the software and required modification on the ONOS Southbound to avoid the creation of flows with the aforementioned instruction. Finally, Feature Requests contains the procedures meant to verify the support for the new functionalities introduced by Corsa. There is only one test to verify the progress of the Q-in-Q encapsulation with EtherType 0x8100 and 0x88a8 [Q-in-Q].

Testing results are summarized in Table 10.1: OF Conformance test results summary, results reported hereafter refers to the last firmware released by Corsa during Phase B.

#	ID	Requirement	Result	Remarks
Flows	1.0	Packet In	Success	
	1.1	Forwarding using MAC addresses	Success	
	1.2	VLAN encapsulation	Success	
	1.3	VLAN translation	Success	
	1.4	VLAN decapsulation	Success	
	1.5	Forwarding to input port	Success	
	1.5.1	Forwarding to input port 1	Success	
	1.6	Forwarding to multiple output ports	Success	
	1.6.1	Forwarding to all ports	Success	
	1.7	Forwarding to multiple output ports (each port as unique VLAN tag)	Fail	FlowMod error
Bug	2.1	Clear deferred	Success	
Feature requests	3.1	QinQ encapsulation	Success	
	3.2	QinQ encapsulation1	Success	
	3.2.1	QinQ encapsulation1 packet in	Success	
	3.2.2	QinQ encapsulation1 packet out	Success	
Groups	4.1	Single Group Bucket type Indirect	Success	
	4.2	Single Group Bucket type All	Success	
	4.3	Multiple outputs with group	Success	

#	ID	Requirement	Result	Remarks
	4.4	Push VLAN on flow and multiple outputs with group	Success	
	4.5	Multiple outputs with push VLAN tag with group	Fail	Groupmod error
	4.6	Push VLAN on flow and outputs with indirect group type	Success	
	4.7	Push VLAN and outputs on bucket of indirect group type	Success	

Table 10.1: OF Conformance test results summary

11 VPLS Tests

VPLS (Virtual Private LAN Service) is an ONOS application that creates L2 broadcast "overlay" networks on-demand, on top of an OpenFlow infrastructure. The application interconnects hosts connected to the OpenFlow data plane using "overlay" broadcast networks. The hosts can send untagged or VLAN tagged traffic, using either the same or different VLAN ids. Figure 11.1 shows an example of VPLS overlay network with five hosts and two VPLS networks; to be noted that hosts do not need to use the same access VLAN but they can send traffic using different tags. VPLS application will assure proper translation at the egress of the network.

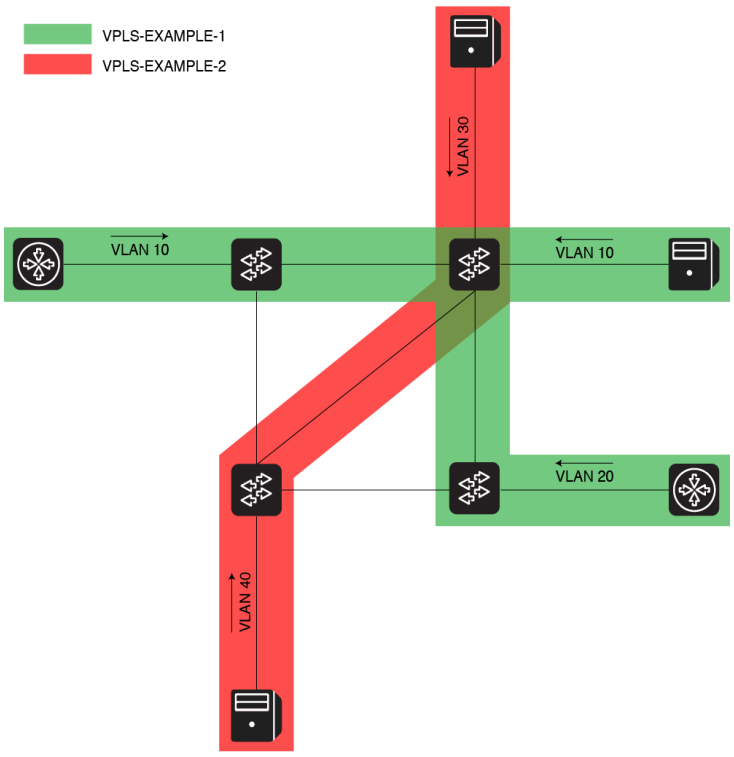

Figure 11.1: Example of VPLS setup

VPLS can be configured using ONOS CLI or pushing a *netcfg* configuration file (by JSON configuration file). Each time a new VPLS needs to be created, we need to specify the endpoints, their parameters (access VLAN for example) and the type of VPLS (*encap* or *noencap* – using encapsulation or not). VPLS application is currently tested on Cambridge Lab facility where two Corsa DP2100 switches directly interconnected and three host VMs are used. A fourth VM is used as ONOS controller running VPLS application. Figure 11.2 shows the testing scenario deployed in the Cambridge Lab.

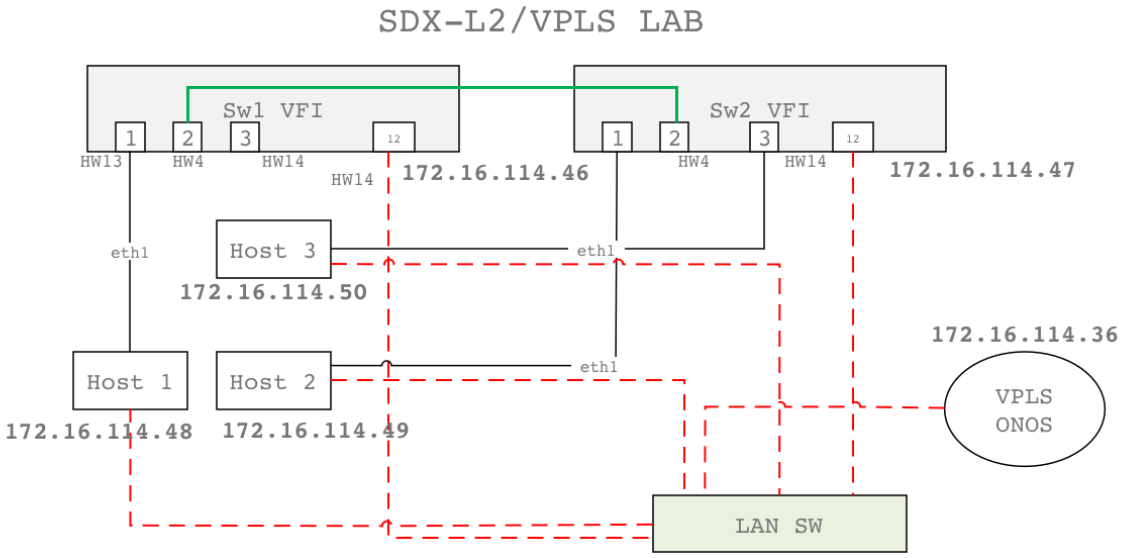

Figure 11.2: VPLS Test scenario on Cambridge Lab

As regards VPLS testing, several tests have been designed to test infrastructure functional and non-functional properties. Tests are classified using 6 categories: i) ONOS data plane; ii) ONOS control plane; iii) VPLS configuration; iv) VPLS mode; v) VPLS Corsa; vi) VPLS performance. First two groups test some ONOS basic functionality that are used by VPLS. For example, the first group considers the ONOS data plane functionality like creating multi-point to single-point virtual circuit and vice versa. Instead, the second group considers some control plane aspects like management of network endpoints. VPLS configuration verifies application functionality using both ONOS CLI or ONOS NETCFG; VPLS mode tests different type of VPLS. Then, there are some Corsa specific tests and there is a group meant to assess the performance of the control plane. Table 11.1 shows all the tests performed during Phase B and their results together with comments and remarks.

#	ID	Requirement	Result	Remarks
ONOS data plane	1.1	sp2mp virtual circuit	Success	
	1.2	sp2mp virtual circuit - VLAN	Success	
	1.3	sp2mp virtual circuit - VLAN encapsulation	Success	
	1.4	sp2mp virtual circuit - VLAN - VLAN encapsulation	Success	ONOS CLI cannot be used for the testing
	1.5	sp2mp virtual circuit - VLAN - VLAN translation	Fail	CORSA devices cannot apply different actions on multiple outputs

#	ID	Requirement	Result	Remarks
	1.6	sp2mp virtual circuit - VLAN - VLAN encapsulation - VLAN translation	Fail	CORSA devices cannot apply different actions on multiple outputs
	1.7	mp2sp virtual circuit	Success	
	1.8	mp2sp virtual circuit - VLAN	Success	
	1.9	mp2sp virtual circuit - VLAN encapsulation	Success	
	1.10	mp2sp virtual circuit - VLAN - VLAN encapsulation	Success	
	1.11	mp2sp virtual circuit - VLAN - VLAN translation	Success	
	1.12	mp2sp virtual circuit - VLAN - VLAN encapsulation - VLAN translation	Success	
ONOS control plane	2.1	Interface commands - Add new interface	Success	
	2.2	Interface commands - Add new interface MAC	Success	
	2.3	Interface commands - Add new interface VLAN	Success	
	2.4	Interface commands - Add new interface MAC+VLAN	Success	
	2.5	Interface commands - Change interface	Success	
	2.6	Interface commands - Remove interface	Success	
	2.7	Interface commands - Lists of interfaces	Success	
VPLS configuration	3.1	CLI Test - Add new VPLS	Success	
	3.2	CLI Test - Remove VPLS	Success	
	3.3	CLI Test - Add Interface	Success	
	3.4	CLI Test - Remove Interface	Success	
	3.5	CLI Test - Lists of VPLS	Success	
	3.6	CLI Test - VPLS details	Success	
	3.7	CLI Test - Clean state of application	Success	

#	ID	Requirement	Result	Remarks
	3.8	NETCFG Test - Add new VPLS	Success	
	3.9	NETCFG Test - Remove VPLS	Success	
	3.10	NETCFG Test - Add Interface	Success	
	3.11	NETCFG Test - Remove Interface	Success	
	3.12	NETCFG Test - Complete VPLS setup	Success	
	3.13	NETCFG Test - Complete VPLS setup: VLAN encapsulation	Success	
	3.14	NETCFG Test - Clean state of application	Success	
VPLS mode	4.1	Different type of VPLS - encap NONE	Success	
	4.2	Different type of VPLS - encap VLAN	Success	
	4.3	Different type of VPLS - encap MPLS	Fail	MPLS not supported.
VPLS Corsa	5.1	Discovery CORSA switches	Success	
	5.2	Check Driver	Success	
	5.3	Bootstrap flows	Success	
VPLS performance	6.1	Create 4 VPLS	Success	
	6.2	Create 8 VPLS	Success	
	6.3	Create 10 VPLS	Success	Some VPLS are wrongly shown in failed state. Intents are correctly installed and pings work.
	6.4	Create 100 VPLS	Success	
	6.5	Create 500 VPLS		Application stuck, intents not installed and pings not working
	6.6	Create 1000 VPLS		Application stuck, intents not installed and pings not working

Table 11.1: VPLS test results summary

Most of the tests were successful. However, we found severe issues during tests at scale. For example, it was not possible to create a bulk of 500 VPLS; during testing, application seemed to be unresponsive, no intents were installed and pings were not working. Additionally, during testing sessions we also faced some strange situations. For example, during test 6.3 VPLS, CLI showed some VPLS in failed state.

At the same time, all the related flows were installed and the pings were working. In general, we can state that most of functionality properly work, but corrections are needed to make VPLS more stable.

In order to automate the execution of the tests, some management scripts have been developed which: i) configure VLAN ID and IP Address in the target hosts; ii) create and push a *netcfg* in ONOS according to the scenario to be tested; finally, iv) check the connection with a series ping between hosts belonging to the VPLSs.

Other tests have been designed and planned but currently they cannot be executed due to limitations of the testing environment. These include control plane resiliency and data plane resiliency tests.

12 Monitoring: Software Testing

During JRA1's development efforts in the testbed environment (GÉANT's Cambridge Lab), troubleshooting, monitoring and testing ONOS Deployments were found to be tedious tasks involving manual operations such as:

- Connecting to different nodes.
- Inspecting and filtering log files.
- Gathering metrics related to the nodes (status, resource consumption, etc.)
- Preserving information and metrics gathered for future reference.

In order to overcome the aforementioned problems, a solution was designed and implemented to facilitate and automate tasks. Typical examples include, but are not limited to:

- Collecting and storing data (e.g. important events, system metrics, logs).
- Visualising collected data and highlighting potential problems.
- Performing queries, as needed.

Figure 12.1 depicts a generic architecture designed and implemented as an indicative working prototype within the Cambridge lab environment. The prototype mainly utilises components from the Elastic Stack. Namely, we use Beats (FileBeat, MetricBeat, PacketBeat) as lightweight shippers for Log Files, System/Network related metrics. Additional information can be collected from the network elements using traditional methods such as SYSLOG, and SNMP. The aforementioned lightweight shippers send all gathered data to a processing Node running Logstash software.

The focus has been mainly on SDN controller log files, CPU utilisation, RAM consumption, disk status, per- flow byte and packet counters (grouped by their L2-L4 headers). In turn, the input data stream is structured and enriched based on a predefined pipeline processing and the output is exported to a datastore. ElasticSearch suited the particular needs of JRA1, and access to the data stored on instances is either via a RESTful API or by using specific tools for performing queries and creating visualisations (e.g. Kibana, Grafana).

Within the SDN-based circuit on demand Use Case, the platform analysed above has been used more as a debugging and less as a monitoring tool. Specifically, queries are performed on the datastore based on instances, log timestamp, and severity of the message. In essence, any information stored in ElasticSearch may be visualised (e.g. CPU/RAM usage as a time series data). Note that the logs shipping procedure to the Portal Webhost instance, and IDM instance has also been extended.

Pending proper configuration, the platform can be tailored to a number of setups, depending on the operational needs/constraints towards providing considerable operational benefits (e.g. troubleshooting a circuit service instance, monitoring the deployment of an SDN-based circuit on demand instance), when used in a production environment.

In Phase A, we considered the following as potential advancements related to monitoring: a) linking specific logs with tenants/their respective service requests, b) multi-source data correlation towards identification of strange events – system problems.

Regarding (a), most logging and monitoring solutions, including the aforementioned, would need to have internal information (e.g. to associate log messages to various services - circuits). Additionally, such functionality would also have an impact on the ingestion/processing performance. Alternatively, this could be integrated within the actual software by adding important information in the log lines. Consequentially these lines could be processed appropriately by the ingestion pipeline, essentially "tagging" the log (e.g. inserting the service name for all relevant logs). Regarding the correlation scenarios, we would need a simple but effective mechanism/interface through which the correlations can be defined (e.g. Conditions/Thresholds, Events). This functionality could be achieved via a separate tool or incorporated within the existing stack. However, we consider (a) as a pre-requisite to (b) since most approaches for correlating log messages would likely require "tagging".

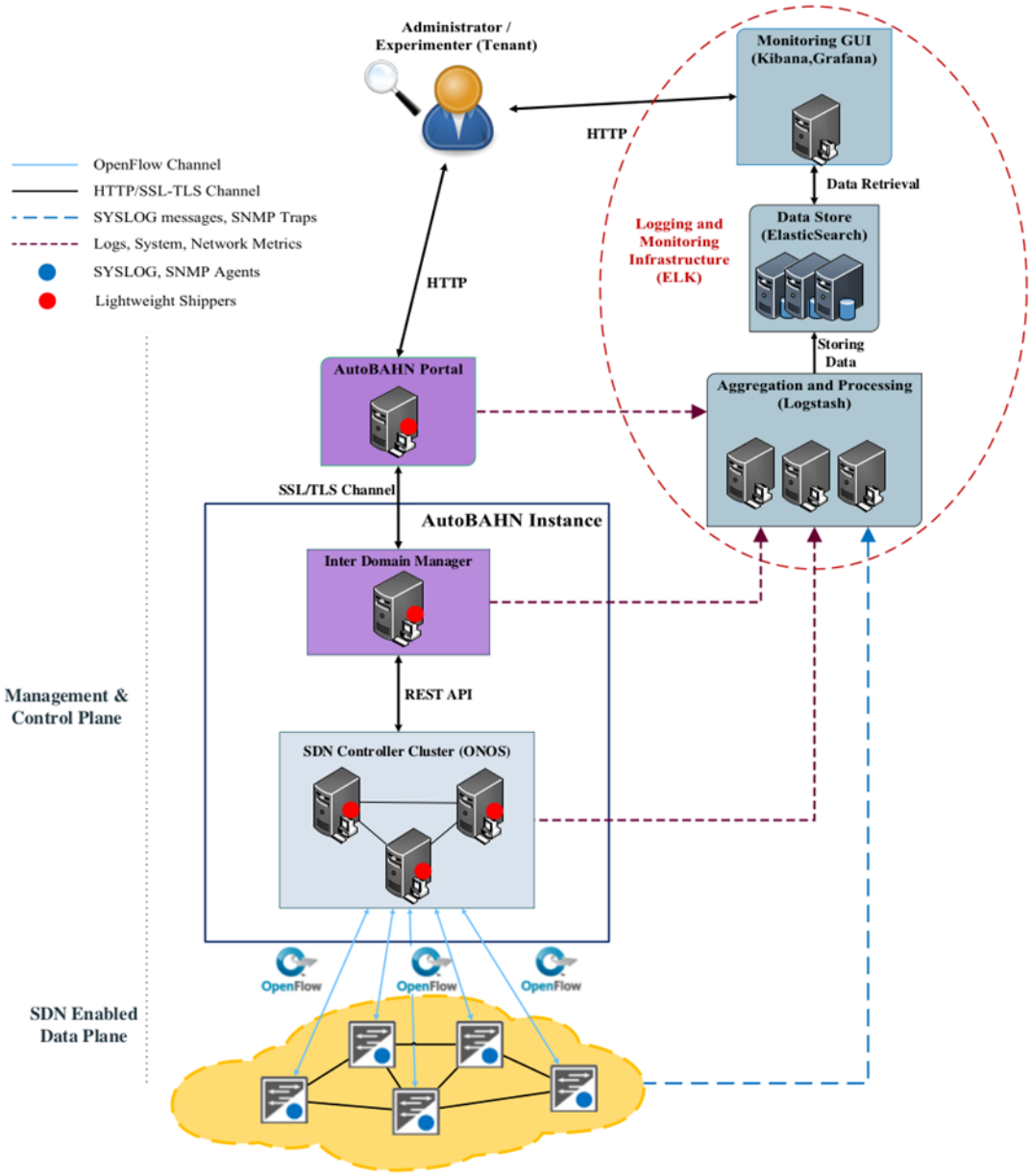

Figure 12.1: Monitoring architecture

Additional information on accessing and using the infrastructure is available to project participants at the JRA1 Monitoring and Infrastructure Wiki.

As part of our efforts during Phase B within JRA1, parallel with software development, we have been continuously performing DevOps-related tasks such as: (a) configuring the infrastructure (test VMs, CORSA switches), (b) integrating/deploying the software in the appropriate instances and (c) functionally testing new software versions for compatibility. As part of automating the above, we developed auxiliary modules to interact with the corresponding components. Moreover, we utilized the Ansible automation framework for streamlining repetitive testing tasks and orchestrating complex actions which would otherwise require additional time and effort to be executed manually.

In general, testing can be performed in a variety of ways, such as: Unit Tests, dedicated frameworks (TestON) and manual operations (logging to different machines, creating services, generating traffic, retrieving the results). Up to this point, our efforts have been focused on manual operations and interactions with the actual infrastructure. Examples of the implemented scenarios include, but are not limited to: (a) configuring the testing environment (e.g. setting up IPs and MTUs), (b) performing data plane functional testing (allocating a network circuit, generating traffic ICMP/IPerf tests, presenting the results), (c) retrieving the flows and meters and (d) interacting with the CORSA devices through their REST API.

13 GTS testbed for SDX L2 testing

During Phase A, SDX-L2 was initially assessed in the Cambridge network to evaluate new features and to perform other tests such as performance and reliability tests. Later on, the evaluation also moved to the production network (pilot), where it was evaluated, as described in Section 3. During this pilot testing process, there was an increasing need for better control of the underlying network and for full configuration of the environment to speed the testing procedure. An environment isolated from other JRA1 pilots that would allow continuous improvements and/or configuration on the defined scenario, would leave more room for troubleshooting.

To this end, in Phase B, the production topology was replicated and mapped to the resources provided by the GTS testbed. Specifically, four Corsa bridges (Par51, Par52, Mil51, Mil52) were defined in a full-mesh fashion, each connected to two hosts – with a total of four end-user machines (hostA, hostB, hostC, hostD). Connected to the four bridges there is an ONOS cluster comprised of three instances (cO1, cO2, cO3); which host the SDX-L2 app and oversee the different switches in Paris and Milan.

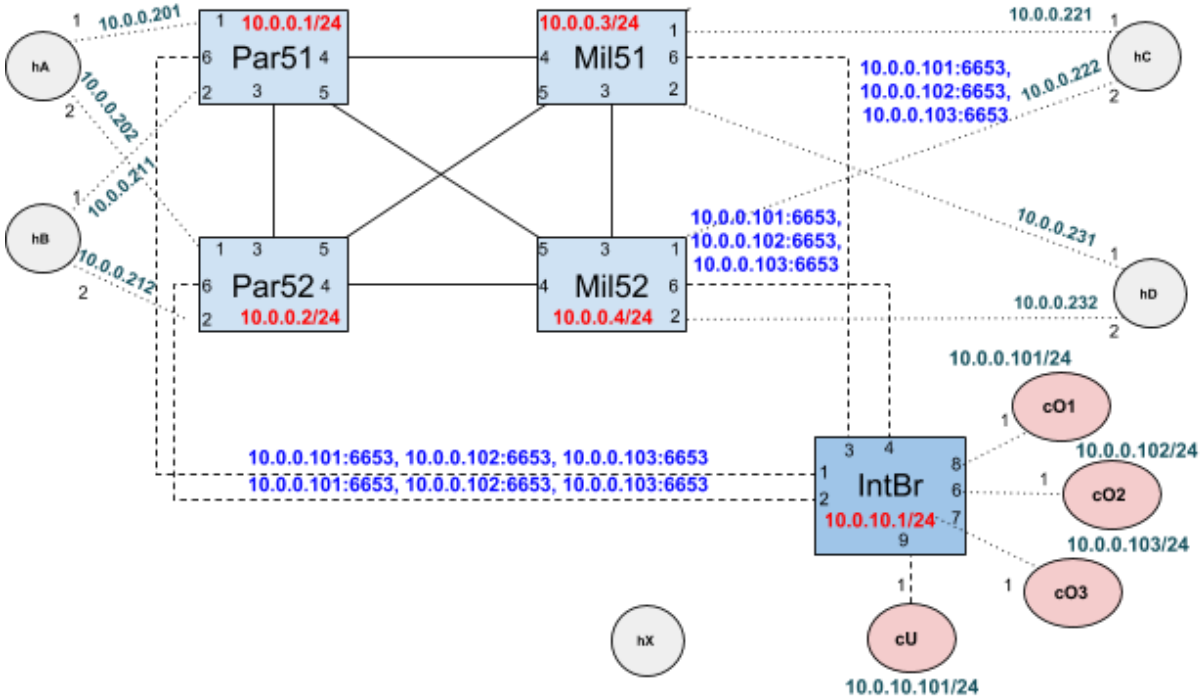

Figure 13.1: SDX-L2 pilot replication in GTS

Due to restrictions in the number of controllers allowed per visual system integrator (VSI) (that is, the "definition" of the bridges within GTS) and the possibility to specify only point-to-point links, it was also necessary to design the management network of the GTS virtual testbed. To this end, an underlying layer was added. This underlying layer realises the management network connecting the management port of the switches (the so called OF port – each VFC has one OF port) to the ONOS instances. An intermediate bridge would connect all the different resources in the overlay: the four bridges and the three instances of the ONOS cluster. GTS does not allow the instantiation of standard bridges in the testbed, therefore an OpenFlow solution was used for the intermediate bridge. A second controller which basically manages the OpenFlow switch of the underlying layer was also deployed.

Two solutions were designed to control this switch: i) ONOS instance running VPLS application, which basically creates a Layer 2 network connecting overlay resources; ii) RYU controller [RYU] running a SimpleSwitch application, which turns the OpenFlow device in legacy Layer 2 bridge.

With this approach, we gained the required control and flexibility to test SDX-L2 in a production-like environment. Connectivity tests were performed to verify basic reachability across the hosts.

During the configuration process and replication (required during transition of the versions within GTS), Ansible scripts were developed to fully automate the set-up of the overlay, as well as to test the basic operations of the SDX-L2 application.

After a minimal configuration – defining the IPs and other credentials provided to each VM in GTS – the scripts are run from a node dedicated to this purpose and with connectivity to all others. Once finished, the ONOS underlay VM must have a single instance deployed and configured appropriately so that the VPLS application can grant connectivity across the networks. The automated tests can then be started to verify the table of SDX-L2 designated tests, amongst others, creation and deletion of instances, connection points and virtual circuits.

14 Conclusion

Technology pilots are very important as early steps of bringing use cases to production, as they provide the necessary operational feedback and real-world insight that cannot be obtained in a development/lab environment. This has been the case with both the SDX L2 pilot and the SDN-based circuit on demand pilot. Operational teams have had hands-on experience with the pilot functionality, trialled features and the operational readiness of both applications to a great extent and provided valuable feedback for their evolution and their maturity. Additionally, the constantly changing codebase of ONOS and related applications provide a basis for evolution and further testing of the pilot functionality. The JRA2 T2 team has made significant contributions to open source software, both via commits to the core ONOS codebase and the SDN applications developed within the Task.

In addition, since monitoring is vital for the operation of a service in production, a monitoring solution has been designed and prototyped, which can be extended in a straightforward manner to more than one use case.

Aside from the technology pilots themselves, emphasis has been placed on verification of the use case functionality and operational readiness upon OpenFlow-enabled equipment from different vendors. For this reason, multiple testbeds in different environments have been utilised. The main goal has been to avoid implicit hardware-specific assumptions and subsequent vendor lock-in. Furthermore, OpenFlow conformance scripts have been developed in order to automate the verification of the required functionality for new switches and controller software upgrades. It is important to bear in mind that results are subject to rapid changes due to software updates and are valid for a limited amount of time.

The experiences and findings will guide the next steps. Based on pilot feedback detailed in this report, focus will be placed upon the SDN-based circuit on demand use case, in order to make sure that the solution is mature enough for user-driven pilots and pre-production deployments.

Appendix A Details of SDN-Based Circuit on Demand Testing in Janet Testbed

A.1 Janet SDN Testbed

The Janet SDN testbed consists of four HP3800 switches located in London, Manchester, Leeds, and Bradley Stoke PoPs, as shown in Figure A.1.

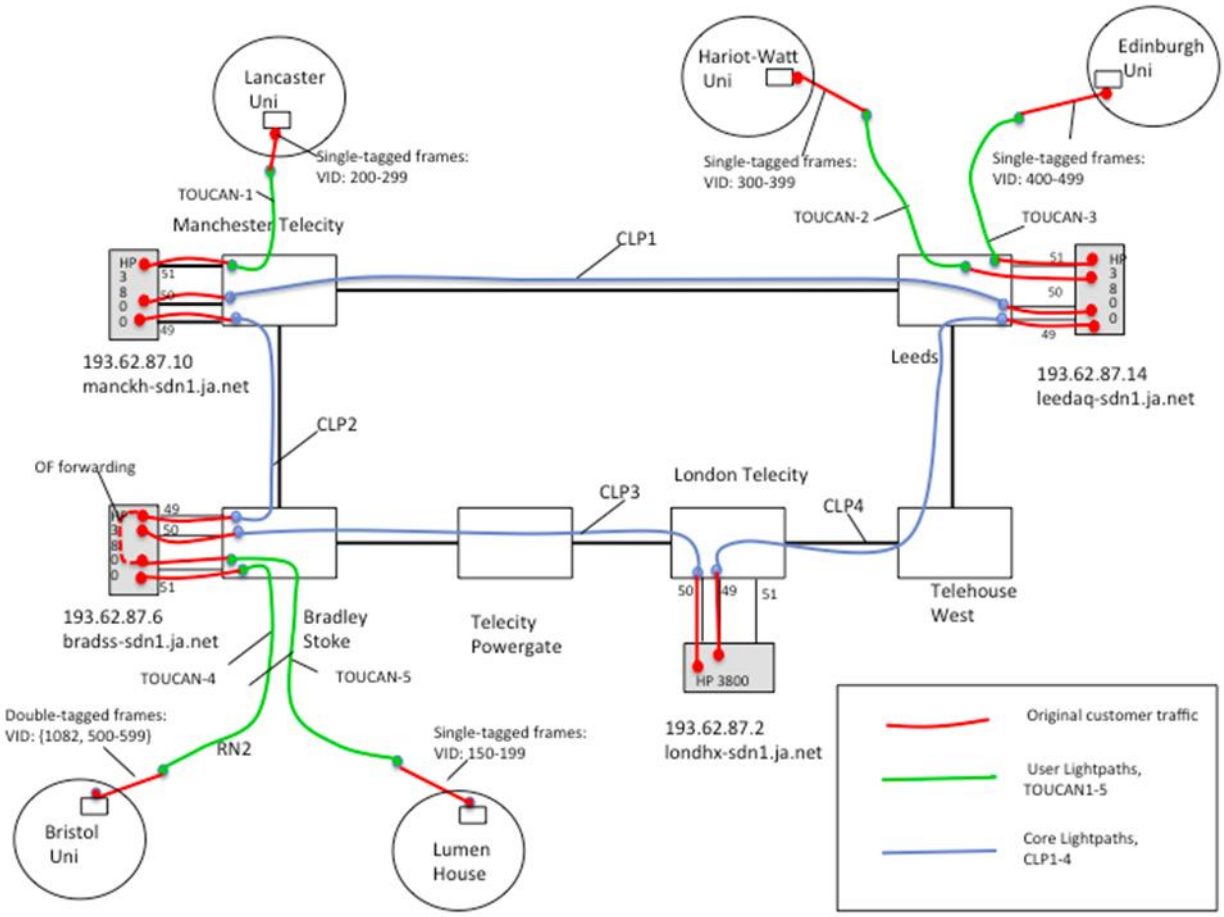

Figure A.1: Janet SDN testbed

The testbed is an overlay network built over the Janet Lightpath infrastructure. It uses EoMPLS circuits of the Janet Lightpath infrastructure to connect switches to user sites (1G circuits to ports 51) and to each other (10G circuits between ports 49 and 50).

The Lightpath circuits are transparent for the testbed traffic, so the HP3800 switches receive frames exactly in the form users send them.

User traffic must be tagged. In order to differentiate users on the shared port 51, each user site is assigned a unique range of VLAN IDs.

The switches are configured to support several virtual OF instances, and each OF instance is bound to a unique set of VLAN IDs and can be controlled by several SDN Controllers.

The testbed virtual instances (slices) are created on a per-test basis. For example, if test A involves two user testbeds, say Lancaster and Lumen House, then instance A should include sets of VLAN IDs of the both users.

To prevent loops and an unwanted LAN-style connectivity, it is recommended to assign ports to VLANs in a non-contiguous way, e.g. if port 51 of London Telecity switch is assigned to VLAN 151, then ports 49 and 50 of that switch should not be assigned to VLAN 151.

A.2 Janet SDN Lab

To verify the fact that DynPaC can work with HP3800 switches, on which the Janet distributed testbed is built, Janet SDN lab testbed was used. It consists of two HP3800 switches located at Lumen House, Oxfordshire (Janet/JISC headquarters).

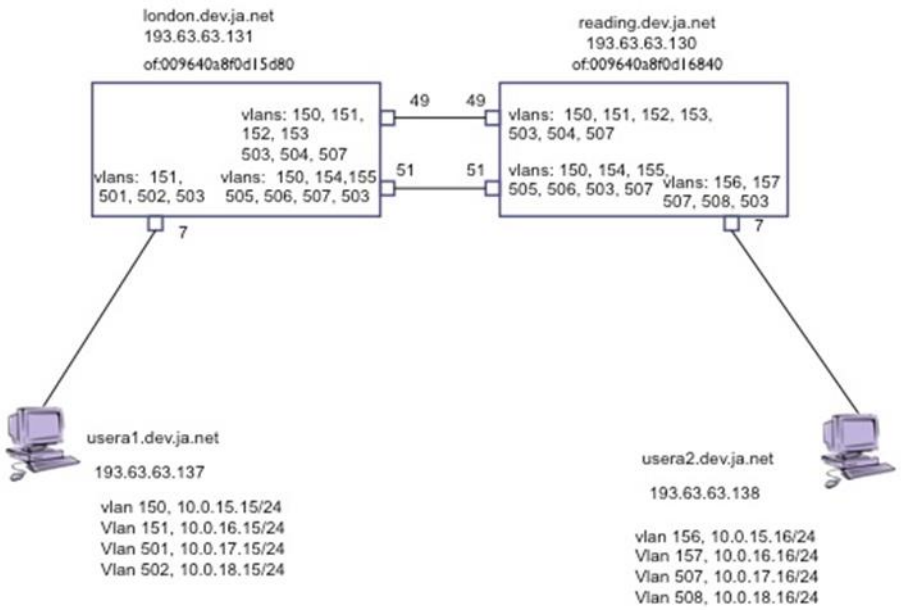

Figure A.2: Janet SDN lab

The switches are interconnected by the two 10G links (ports 49 and 51) to create a physical loop, and hence, to emulate the ring topology of the Janet SDN testbed as close as possible. There are two hosts connected to port 7 of the switches that emulate user sites (Figure A.2).

The SDN lab configuration has been tested and the tests objectives were:

- To provide a connectivity between the hosts interfaces sitting on different VLANs.
- To check whether a drop meter limits traffic according its rate parameter.

Both tests brought the positive result but only after some changes in both ONOS and DynPaC have been made. Some problems still had to be solved, so we had to change the configurations of the HP3800 switches to make them accept some DynPaC rules and support ONOS link auto-detection functionality. These configuration changes are reflected in the lab testbed diagram above:

- VLAN 150 is assigned to all the ports on both switches to support ONOS BDDP auto-detection.
- VLAN 503 is assigned to all the ports on both switches to test the scenario without the DynPaC VLAN translation function.
- VLANs 501 and 507 are assigned to the ports in a way that corresponds to the current DynPaC limited VLAN translation technique.

When the remaining problems (described in detail below) are solved, the testbed config will be changed to be in line with the VLAN assigning rule (non-contiguous).

A.3 Interconnection of Janet testbeds

To test multi-domain features in Janet Testbed, Janet SDN Testbed and Janet SDN Lab were organized as two independent domains. Namely, separate ONOSes and corresponding IDMs were installed and interconnection was provided by Juniper MX80 as shown in the following Figure A.3.

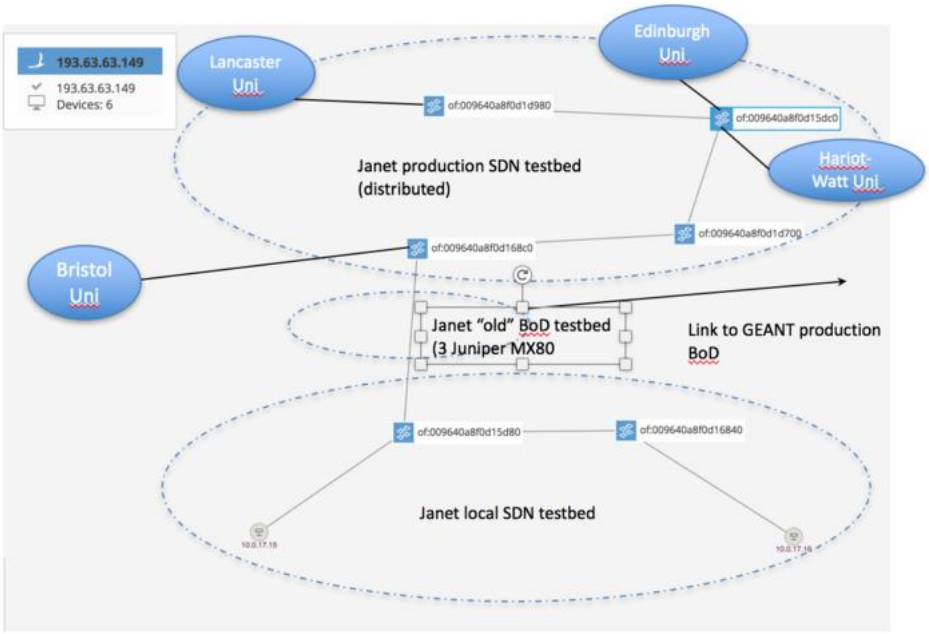

Figure A.3: Interconnection of Janet domains

Testing results were as expected:

- Multi-domain service can be provisioned through Janet lab testbed using AutoBAHN GUI.
- VLAN translation uses intermediate VLAN ID on the inter-domain switch links.

A.4 ONOS Support for HP3800 Switches

Up to ONOS 1.8.x, the ONOS default driver was patched with the changes developed by Lancaster University.

In ONOS 1.9.0, PacketService was significantly changed and all APPLY_ACTIONS were accompanied with CLEAR_ACTIONS instruction in FLOW_MOD messages to prevent sending a packet to the controller and generating its copy to data plane in TTP scenario. For example, assume the packet has a match in table i-1 and apply_actions are performed. An action_group is set in the write_actions and there is goto_table to i instruction. In table i the packet matches again, output_to_controller is performed and the write_actions in i-1 are performed, which also generates a copy of the packet in the data plane.

HP3800 switches do not support the CLEAR_ACTIONS type of instruction described, instead, they reject all the handshake FLOW_MOD messages. As a result, it was decided to develop a full ONOS HP3800 driver instead of patches to solve CLEAR_ACTIONS and some other issues.

Adhering to the ONOS principle that a driver should handle special device characteristics, the JRA1 team developed an ONOS driver for HP3800 switches based on the Lancaster University patches. The

driver does not add CLEAR_ACTIONS (clear deferred in ONOS terminology) instruction in FLOW_MOD messages and also solves some other issues, such as:

- Filtering ETH_TYPE=VLAN criterion (not supported by HP3800).
- Filtering SET_QUEUE actions (not supported by HP3800).
- Correcting the order of actions in DynPaC FlowObjectives.

Another problem solved was the flow priorities installed by DynPaC. DynPaC generated flow rules with lower priorities than ONOS default controller apps (ARP, LLDP, BDDP). Due to the lower priorities, user packets did not match DynPaC rules and were instead sent to the ONOS controller, according to the default rules. Modification was made to DynPaC to prioritise DynPaC flows.

Starting from ONOS, 1.13, the ONOS community changed default driver for HP switches. Now it installs rules onto the software table instead of to the hardware one. As software table is slower, modifications which will make default table configurable are considered.

A.4.1 Actions Order in DynPaC FlowObjectives

Up until ONOS 1.8.0, when the patched default driver was used, DynPaC had to take care of the order of actions, i.e. to specify the VLAN ID first and then the output port. The developed HP3800 driver solved this problem with correct FlowObjective to FlowRule translation, resulting in the proper actions order.

A.4.1.1 *BDDP Link Discovery*

ONOS uses LLDP and BDDP packets to detect network infrastructure links. The tests showed that ONOS could not detect links between HP3800 switches using BDDP when switch ports were not assigned to the first VLAN ID of a OF instance (VLAN 150 in the testbed config). This happened because ONOS injects untagged BDDP packets to packet_out messages, and then HP3800 switches forward BDDP packets tagged with the first VLAN from the instance range. As for LLDP packets, they are not propagated by HP3800 switches, even when a port supports the first VLAN from the instance range.

There are a number of solutions to improve link detection:

- Short-term solution: all ports of two switches are assigned to VLAN 150. As there is a logical loop on VLAN 150, this VLAN is not used for user reservations. This solution was tested and proved successful.
- Long-term solution 1: to generate BDDP packets tagged with VLAN IDs supported by switch ports, as LLDP–based detection does not work at all on HP3800 switches. To implement this solution, new ONOS CLI and REST API commands will be implemented. It will generate (during the specified period of time or specified number of packets) BDDP packets with the specified VLAN ID tag and send these packets in packet out messages though specified port or ports, as different ports might be bound to different VLANs.
- Long-term solution 2: to edit the topology manually, correcting the auto-detected topology. This solution will be implemented in a separate ONOS application. It will be enabled to read the topology information from a JSON file. However, the remaining problem is how to automatically track topology changes.

A combination of the two, long-term solutions looks the most promising.

A.4.1.2 *VLAN Translation*

Prior to improvements of the DynPaC translation technique, it only took place at the switch egress. DynPaC was not able to take into account whether intermediate ports (in the same DynPaC domain) along the path supported some VLANs or not. The short-term solution, used before DynPaC VLAN translation technique was made more flexible, was to configure VLANs on switches in a way that fits the DynPaC translation technique. In the more recent DynPac version, this drawback has been addressed, and VLAN translation can now take place in any place in the network, as required.

Appendix B Details of SDN-Based Circuit on Demand Deployment in EHU Testbed

B.1 Introduction and Context

The aim of this section is to present the SN4I (SmartNetworks for Industry) infrastructure, which will be used to deploy services that spawn from technical offices and research labs to an advanced manufacturing facility (the AAMC), interconnecting real machine tools in order to allow experimentation in Industry 4.0 leveraging NFV (Network Function Virtualization) [ONF] and SDN [ETSI-NFV] technologies.

Recently, the University of the Basque Country UPV/EHU, with companies', local, and autonomous government financial support has launched the AAMC (Aeronautics Advanced Manufacturing Centre) [AAMC]. This infrastructure accounts for €9 M investment that will reach €16 M in 7 years. The aim of the AAMC is to develop advanced manufacturing technologies in developments levels that allow fast transfer to industrial applications, key to the economy of the Basque Country, the region where the AAMC is located. Therefore, its R+D+I activity is focused on TRL 5 to 7, which is often referred in the innovation industry as the "Valley of Death", due to it being the point from which many new ideas fail to progress.

Additionally, the AAMC also serves as a meeting point and a location that boosts cooperative work among different agents and companies with diverse capacities, interests and trajectories in the sector of aeronautics and structural engine parts. More specifically, the AAMC aims to provide a mixed R+D+I centre in advanced manufacturing technologies that will become a reference at international level. In order to achieve this goal, the AAMC owns a manufacturing plant with real equipment, with the implications it conveys from the point of view of experimentation deployment in manufacturing processes. Figure B.1 provides a graphical view of the AAMC plant.

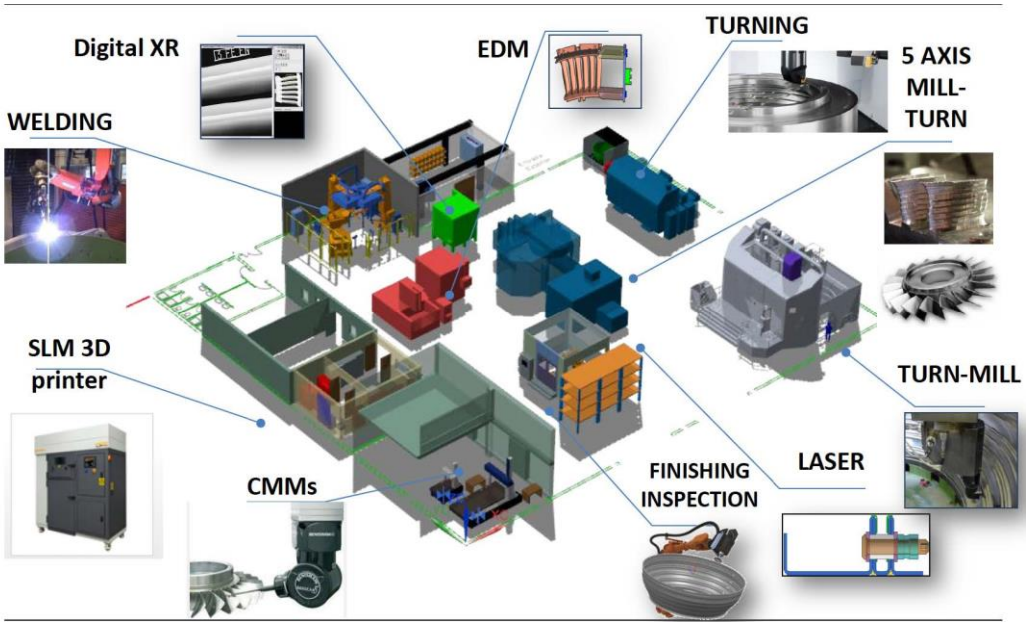

Figure B.1: Schema of the AAMC plant

Regarding experimentation on SDN and NFV technologies, it is worth noting that the UPV/EHU has operated one of the first OpenFlow experimentation networks, named EHU-OEF [EHU-OEF]. This network has been used to test novel network virtualisation schemas (L2PNV) and network authentication and access control systems (FlowNAC) and has allowed the deployment of new services (EHU-OEF Lightweight Unify Domain). However, EHU-OEF has fulfilled its lifecycle, and the deployment of its successor is ongoing: SN4I (SmartNetworks for Industry), which is graphically depicted in Figure B.2. The goal of SN4I is to provide a flexible network to allow experimentation on new paradigms in the manufacturing process and within the manufacturing plant itself. This infrastructure, which will start working in the summer of 2018, will interconnect the research laboratories of the Faculty of Engineering in Bilbao, the presence point of RedIRIS (the Spanish REN) and I2Basque (the Basque REN) in the UPV/EHU and the AAMC. For this aim, both the networks of the UPV/EHU and I2Basque will be used.

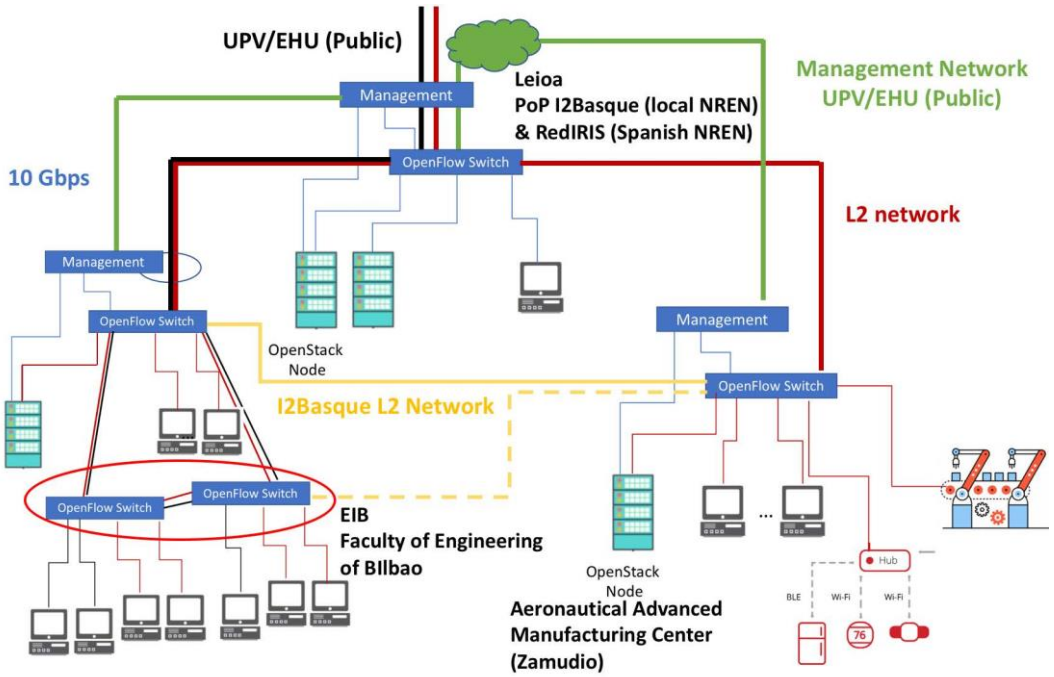

Figure B.2: SN4I schema

B.2 SN4I Experimental Facility Implementation Details

The SN4I infrastructure consists of an OpenFlow-based Layer 2 network, which will support the deployment of Industry 4.0 services. For this aim, NFV technologies will be used and more specifically, the ETSI OSM orchestrator over OpenStack (a very popular Open Source Virtual Infrastructure Manager [OPENSTACK]) and OpenVIM [OPENVIM]-based VIMs will be used.

As it can be seen in Figure B.3, the SN4I network has been designed to interconnect three geographically distant locations: the University of the Basque Country's Faculty of Engineering of Bilbao, the University of the Basque Country's headquarters in Leioa (about 10 Km from Bilbao) and the AAMC in Zamudio (about 12 Km from Bilbao and about 17 Km from Leioa). In each of these locations, a virtual infrastructure node is deployed, managed by the OpenStack VIM. Each OpenStack node consists of four physical servers: one of them is configured as the controller, where the different modules necessary to manage and configure the virtual infrastructure reside; and the other three are configured as computing nodes. These three nodes provide a pool of resources which the actual virtual services (VNFs) will use for their execution. The controller node is the one responsible for the distribution of virtual resources to the virtual services, namely the mapping between virtual and physical resources and the implementation of isolation and security functionalities among different virtual services running over the same physical hardware.

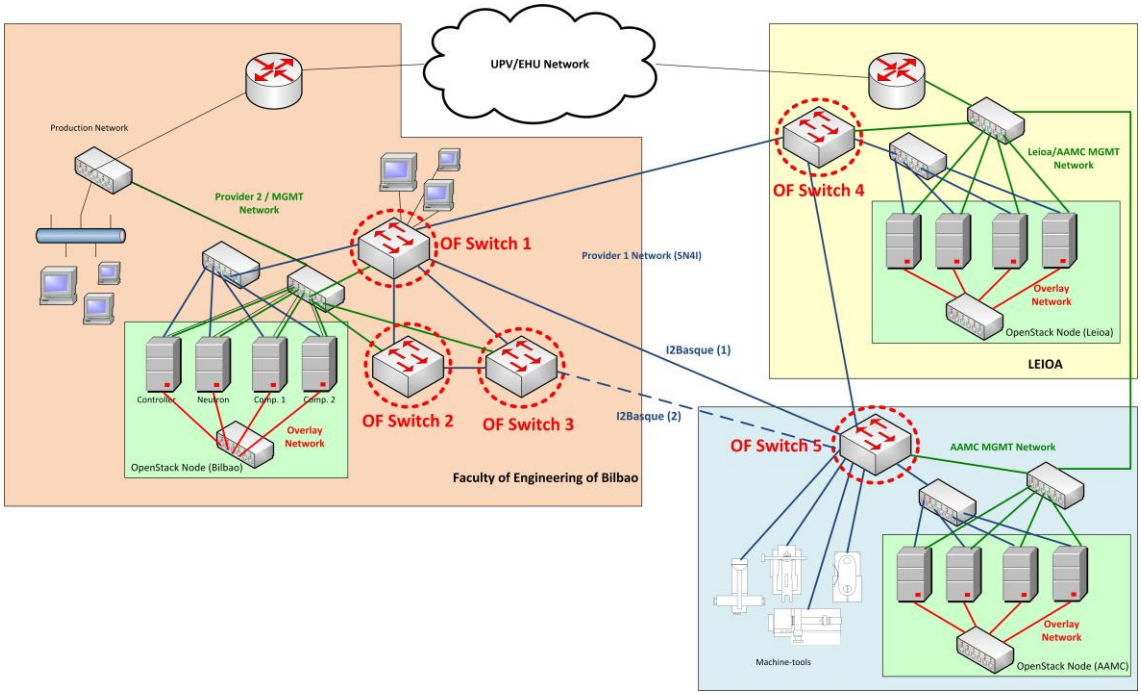

Figure B.3: SN4I experimental facility network architecture

The three OpenStack nodes deployed in the three, separated locations are then interconnected at Layer 2 and at a data rate of 10 Gbps by means of fibre optic links and a set of OpenFlow switches (blue lines in Figure B.3). This network is known as the provider network, in terms of OpenStack terminology. In fact, currently the SN4I network consists of five OpenFlow switches (three in Bilbao, one in Leioa and one in the AAMC in Zamudio) interconnected forming a ring. The interconnection among the three switches in Bilbao and between this location and Leioa is carried out thanks to fibre optic links provided by the University of the Basque Country. Then, the OpenFlow switch in Leioa is connected to the corresponding switch in the AAMC by a link provided by the University of the Basque Country and the ring is closed by a connection provided by I2Basque, the Basque REN (Research and Education Network), between the AAMC and the Faculty of Engineering of Bilbao. It is even possible that in the future, I2Basque will provide a second link to interconnect a second OpenFlow switch in the Faculty of Engineering of Bilbao with the AAMC. This possible link is marked with a dashed line in Figure B.3. The aim of deploying this second connection between the Faculty of Engineering of Bilbao and the AAMC is to achieve a more complex architecture in which redundant paths can be provided.

The management of OpenStack nodes and OpenFlow switches is performed out-of-band. That is, instead of using the deployed Layer 2 experimental network also for management and configuration duties, a separated network used for this goal. More specifically, the Layer 3 network of the University (marked in green in Figure B.3) is used for this aim. In this regard, it is worth noting that all the OpenStack nodes in the premises of the Faculty of Engineering of Bilbao are connected to a second provider network, marked in black in Figure B.3. This second provider network is used to publish internal services of our research group (such as DHCP, NTP, etc) not related to the experiments with the AAMC.

Additionally, it can be noted that in each OpenStack node, all the physical servers that constitute the node are interconnected among them (red links in Figure B.3). This internal network used for the

interconnection of the servers within an OpenStack node is known as the overlay network and it is used for distribution of virtual services to physical and virtual infrastructures, security and isolation related features, etc. These nodes support high performance network connections through the use of technologies like Passthrough and SR-IOV.

Finally, the three OpenStack nodes are managed by a NFV MANO located in the premises of the Faculty of Engineering of Bilbao. In this way, it is possible to create end-to-end virtual services that may consist of different virtual services that could run in any of the three OpenStack nodes, taking into account network and hardware resources efficiently. Additionally, in order to create the virtual links that will interconnect the different virtual services potentially running in any of the OpenStack nodes, the MANO communicates with a SDN controller, situated also in the Faculty of Engineering of Bilbao.

This deployment will allow to study the integration of these NFV technologies with the Industrial Internet protocols and the coexistence with other TSN-based networks. Additionally, in order to compute optimal paths to route the traffic between the different endpoints connected to the SN4I infrastructure, the dynamic and resilient DynPaC framework will be used as a WIM (Wide Area Infrastructure Manager) under ETSI OSM, as it is shown in Figure B.4.

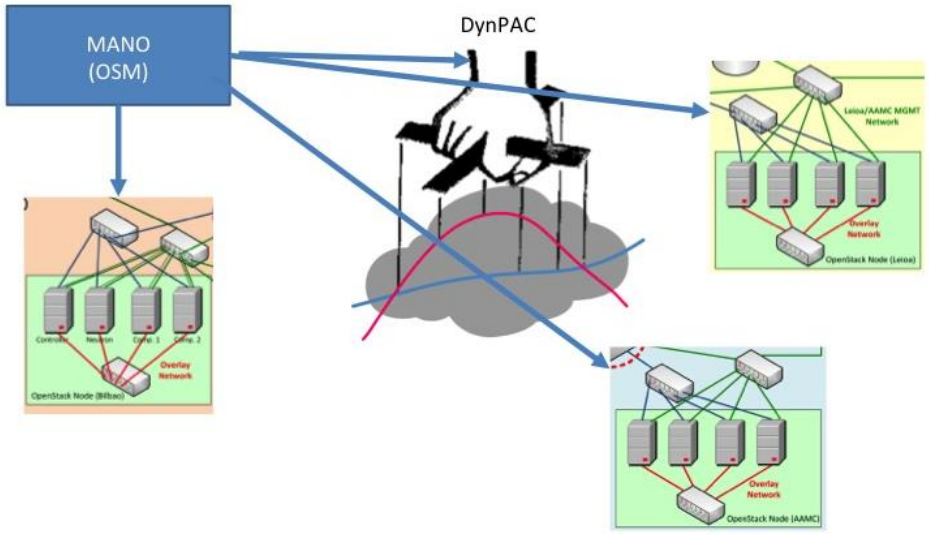

Figure B.4: Integration of ETSI OSM and DynPaC WIM

As described in detail in Section 4, DynPaC is a bandwidth on demand service that, given a source and destination port, it calculates the shortest possible path between these two points with the desired bandwidth guaranteed, taking into account the network state. Additionally, DynPaC integrates calendaring functionalities, which allows to program services to start and end at a given time in the future. Moreover, another interesting feature of DynPaC is that it provides fault-tolerant connections by means of the computation of a secondary backup path which is used to transmit the traffic of the affected service in the case of failure in the primary path. This will be the mechanism to create end-to-end isolated virtual links for each service, with a guaranteed bandwidth and with the possibility of providing a redundant path.

As a result, SN4I will provide an infrastructure where several isolated services can be deployed at the same time, using network, computing and storage slicing and involving specific machines and their

corresponding services. These slices will be able to support most experiments (a notable exception at the time of writing are Real Time protocols which need deterministic timing). It is also interesting to note that deploying a SDN infrastructure can help implementing advanced services like Flow Authentication or attack detection and mitigation.

SN4I will complement traditional research and activity in the AAMC field of manufacturing, with state of the art technologies like high bandwidth, low delay, on demand service creation, SDN-based security, etc. The integration of all these technologies with cutting-edge machine tools and manufacturing processes will allow to provide a mixed R+D+I centre in advanced connected manufacturing technologies that will become a reference at international level.

References

[AAMC]	Aeronautics Advanced Manufacturing Center https://www.ehu.eus/en/web/CFAA/home
[ACINO]	ACINO project http://www.acino.eu/
[AutoBAHN]	https://geant3.archive.geant.org/service/autobahn/pages/home.aspx
[BIG_SWITCH]	http://www.bigswitch.com/
[BRIGADE]	https://wiki.onosproject.org/display/ONOS/Deployment+brigade
[CE]	Carrier Ethernet ONOS application, https://wiki.onosproject.org/display/ONOS/Carrier+Ethernet+Application
[D7.1]	https://www.geant.org/Projects/GEANT_Project_GN4/deliverables/D7.1_Overview-of-SDN-Pilots-Description-and-Findings_Part-A.pdf
[DynPaC]	DynPaC ONOS application, https://wiki.onosproject.org/display/ONOS/BoD+DynPac+application
[EHU-OEF]	UPV/EHU OpenFlow Enabled Facility, http://i2t.ehu.eus/resources/ehu-oef
[GN4-1 POC]	Deliverable D14.1 Future Services PoCs - Implementation and Dissemination, GN4-1, April 2016
[GOPEN]	GÉANT Open Service, https://www.geant.org/Services/Connectivity_and_network/Pages/GEANT_Open.aspx
[GTS]	GÉANT Testbeds Service, https://www.geant.org/Services/Connectivity_and_network/GTS
[ICONA]	ICONA project, https://wiki.onosproject.org/pages/viewpage.action?pageId=4162523
[ETSI-NFV]	Network Functions Virtualization Introductory White Paper, ETSI NFV ISG, https://portal.etsi.org/nfv/nfv_white_paper.pdf
[NSI]	NSI Connection Services v2.0 - Open Grid Forum https://www.ogf.org/documents/GFD.212.pdf
[OFTEST]	OFTest homepage, http://www.projectfloodlight.org/oftest/
[ONF]	Open Networking Lab, now known as the Open Networking Foundation following a recent merger https://www.opennetworking.org/
[ONOS]	Open Network Operating System, http://onosproject.org/
[ONOS-BRIGADE]	ONOS Deployment Brigade, https://wiki.onosproject.org/display/ONOS/Deployment+brigade
[ONOS-CLUSTER]	https://wiki.onosproject.org/display/ONOS/Forming+a+cluster
[OPENSTACK]	The OpenStack Project, https://www.openstack.org/software/
[OPENVIM]	OpenVIM, https://www.sdxcentral.com/projects/openvim/

[P4] The language for programming the data plane of network devices: http://p4.org/

[PO] Packet Optical ONOS use case, https://wiki.onosproject.org/display/ONOS/Packet+Optical+Convergence

[Q-in-Q] http://www.cisco.com/en/US/docs/ios/lanswitch/configuration/guide/lsw_ieee_802.1q.html

[RYU] Ryu component-based software defined networking framework, https://osrg.github.io/ryu/

[SDN-IP] SDN-IP ONOS application, https://wiki.onosproject.org/display/ONOS/SDN-IP

[SDXL2] SDX-L2 ONOS application, https://wiki.onosproject.org/display/ONOS/SDX-L2+application
SDX L3 ONOS application, https://wiki.onosproject.org/display/ONOS/SDX-L3+application

[TESTON] https://wiki.onosproject.org/display/ONOS/System+Testing+Guide

[VPLS] VPLS ONOS application, https://wiki.onosproject.org/display/ONOS/Virtual+Private+LAN+Service+-+VPLS

Glossary

AAMC	Aeronautics Advanced Manufacturing Centre
ARP	Address Resolution Protocol
AS	Autonomous System
BDDP	Broadcast Domain Discovery Protocol
BGP	Border Gateway Protocol
BoD	Bandwidth on Demand
CE	Carrier Ethernet
CLI	Command Line Interface
CP	Cooperation Protocol
CMDB	Configuration Management Database
DTN-X	Infinera's next-generation multi-terabit transport network platforms
DynPaC	Dynamic Path Computational Framework
EBGP	Exterior Border Gateway Protocol
EHU	Euskal Herriko Unibertsitatea (University of the Basque Country)
EoMPLS	Ethernet over MPLS
ETSI	European Telecommunications Standards Institute
FIB	Forwarding Information Base
GUI	Graphical User Interface
IBGP	Internal Border Gateway Protocol
ID	Identity
L2	Layer 2
L3	Layer 3
LACP	Link Aggregation Control Protocol
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
MANO	Management and Organisation
MEF	Metro Ethernet Forum
MP2SP	MultiPoint to Single Point
MPLS	Multiprotocol Label Switching
MX	Juniper series of Ethernet routers and switches
NDP	Neighbour Discovery Protocol
NFV	Network Function Virtualisation
NOC	Network Operations Centre
NSI	Network Service Interface
OF	OpenFlow
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OTS	Open Transport Switch

OXP	Open eXchange Points
PCE	Path Computational Element
PXM	Packet Switching Module
PO	Packet Optical
POC	Proof of Concept
PoP	Point of Presence
REST	Representational State Transfer
SA	Service Activity
SBI	SouthBound Interface
SDN	Software Defined Networking
SDX	SDN eXchange point
SN4I	SmartNetworks for Industry
SNMP	Simple Network Management Protocol
STP	Spanning Tree Protocol
VC	Virtual Circuit
VFC	Virtual Forwarding Contexts
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPLS	Virtual Private LAN Service
VSI	Visual System Integrator
WIM	Wide Area Infrastructure Manager